Mastering Unlabeled Data

Vaibhav Verdhan

MEAP

Mastering
Unlabeled
Data

Vaibhav Verdhan

MEAP

Mastering Unlabeled Data MEAP V05

MEAP Edition

Manning Early Access Program

Mastering Unlabeled Data

Version 5

Copyright 2022 Manning Publications

To comment go to liveBook

https://livebook.manning.com/#!/book/mastering-unlabeled-data/discussion

For more information on this and other Manning titles go to

manning.com

welcome

Thank you for purchasing the MEAP for *Mastering Unlabeled Data*.

This is a cliché in today's world – *"Data is the new oil and electricity."* Businesses need to analyze patterns and trends of data, detect anomalies, reduce complexity of really high-dimensional datasets and then make sound decisions. There is an ever-growing need to draw meaningful insights which are sometimes quite difficult for us to comprehend. This book is an attempt to equip you with unsupervised learning techniques which will perform the complex modelling for you.

Throughout the book, we introduce an algorithm, examine mathematical and statistical foundation and study the various forms and types. This book is a step to bridge the gap between complex mathematical and statistical concepts and pragmatic real-world case studies. Real-world cases on retail, telecommunication, banking, manufacturing etc. are discussed at length to make the knowledge complete. Python implementation of the datasets completes the knowledge.

We are exploring clustering methods, dimensionality reduction methods and advanced concepts of machine learning. You will also work on text and images along with structured datasets.

We are examining the best practices to be followed and the common issues faced. We are also dealing with end-to-end model development including model deployment. You will develop thorough knowledge and understanding of unsupervised learning based algorithms.

This book is for both budding and experienced data analysts and data scientists. A researcher or a student who wishes to explore unsupervised algorithms or a manager who yearns to feel confident when discussing with their clients are some of the target minds.

A curious mind and an attitude to conquer a mountain is required. To get the

most benefit from this book, you'll need a basic understanding of Python and Jupyter notebook. Some basic understanding of data and data science might prove handy. We are working on real-world datasets which are available freely online.

Please let me know your thoughts in the [liveBook Discussion forum](#) on what's been written so far and what you'd like to see in the rest of the book. Your feedback will be invaluable in improving *Mastering Unlabeled Data*.

Thanks again for your interest and for purchasing the MEAP!

—Vaibhav Verdhan

In this book

1 Introduction to machine learning

"There are only patterns, patterns on top of patterns, patterns that affect other patterns. Patterns hidden by patterns. Patterns within patterns– Chuck Palahniuk"

We love patterns. Be it our business or our life, we find patterns and (generally) tend to stick to them. We have our preferences of groceries we buy, telecom operators and calling packs we use, news articles we follow, movie genre and audio tracks we like – these all are examples of patterns of ours preferences. We love patterns, and more then patterns we love finding them, arranging them and may be getting used to them!

Then there is a cliché going on - "Data is the new electricity". Data is indeed precious, nobody can deny that. But data in its purest form will be of no use. We have to clean the data, analyse and visualise it and then we can develop insights from it. Data sciences, machine learning and artificial intelligence are helping us in uncovering these patterns – so that we can take more insightful and balanced decisions in our activities and business.

In this book, we are going to solve some of such mysteries. We will be studying a branch of machine learning referred to as Unsupervised Learning. Unsupervised learning solutions, are one of the most influential approaches which are changing face of the industry. They are utilized in banking and finance, retail, insurance, manufacturing, aviation, medical sciences, telecom and almost every sector.

Throughout the book, we are discussing concepts of unsupervised learning - the building blocks of algorithms, their nuts and bolts, background processes and mathematical foundation. The concepts are examined, best practices are studied, common errors and pitfalls are analysed and a case study based approach complements the learning. At the same time, we are developing actual Python code for solving such problems. All the codes are accompanied by step-by-step explanation and comments.

This book is divided into three parts. First part explores the basics of unsupervised learning and covers easier concepts of k-means clustering, hierarchical clustering, principal component analysis etc. This part gently prepares you for the journey ahead. If you are already well-versed with these topics, you can directly start with second part of the book. Though, it is advisable to give the chapters a quick read to refresh the concepts.

The second part is at an intermediate level. We start with association rules algorithm like apriori, ECLAT and sequence rule mining. We then increase the pace and study more complex algorithms and concepts – spectral clustering, GMM clustering, t-SNE, multidimensional scaling (MDS) etc. And then we work on text data in the next chapter.

The third and final part is advanced. We are discussing complex topics like Restricted Boltzmann Machine, autoencoders, GANs etc. We also examine end-to-end model development including model deployment, best practices, common pitfalls in the last chapter of the book.

By the time you finish this book, you will have a very good understanding of unsupervised technique based machine learning, various algorithms, mathematics and statistical foundation on which the algorithm rests, business use cases, Python implementation and best practices followed.

This book is suitable for students and researchers who want to generate in-depth understanding of unsupervised learning algorithms. It is recommended for professionals pursuing data science careers who wish to gather the best practices followed and solution of common challenges faced. The content is well suited for managers and leaders who intend to have a confidence while communicating with teams and clientele. Above all, a curious person who intends to get educated on unsupervised learning algorithms and develop Python experience to solve the case studies is well suited.

It is advisable that you have a basic understanding of programming in object-oriented languages like C++, Java, Objective-C etc. We are going to use Python throughout the book, so if you are experienced with Python it will surely help. Basic understanding of mathematics and geometry will help in visualising the results and some knowledge of data related use cases will help to relate to the business use cases. Most important of all, an open mindset to

absorb knowledge is necessary throughout the chapters in the book.

The first chapter is designed to introduce the concepts of machine learning to you. In this opening chapter, we are going to cover the following topics:

1. Data, data types, data management and quality
2. Data Analysis, Machine Learning, Artificial Intelligence and Deep Learning
3. Nuts and bolts of Machine Learning
4. Different types of machine learning algorithms
5. Technical tool kit available
6. Summary

Let's first understand the smallest grain we have – "data" as the first topic. Welcome to the first chapter and all the very best!

1.1 Data, data types, data management and quality

We are starting with the protagonist of everything called "*data*". Data can be termed as facts and statistics which are collected for performing any kind of analysis or study. But data has its own traits, attributes, quality measures and management principles. It is stored, exported, loaded, transformed and measured.

We are going to study all of it now- starting with the definition of data. Then we will proceed to different types of data, their respective examples and what are the attributes of data which make it useful and of good quality.

1.1.1 What is Data

"DATA" is ubiquitous. You make a phone call using a mobile network – you are generating data. You are booking a flight ticket and hotel for upcoming vacation – data is being created. Making a bank transaction, surfing social media and shopping websites online, buying an insurance policy or buying a car – everywhere data originates. It is transformed from one form to another, stored, cleaned, managed and analysed.

Formally put - data is a collection of facts, observations, measures, text, numbers, images, videos. It might be clean or unclean, ordered or unordered, having mixed data types or completely pure and historical or real-time.

Figure 1-1 How we can transform raw data to become information, knowledge and finally insights which can be used in business to drive decisions and actions


Raw Data   Information   Knowledge   Insights
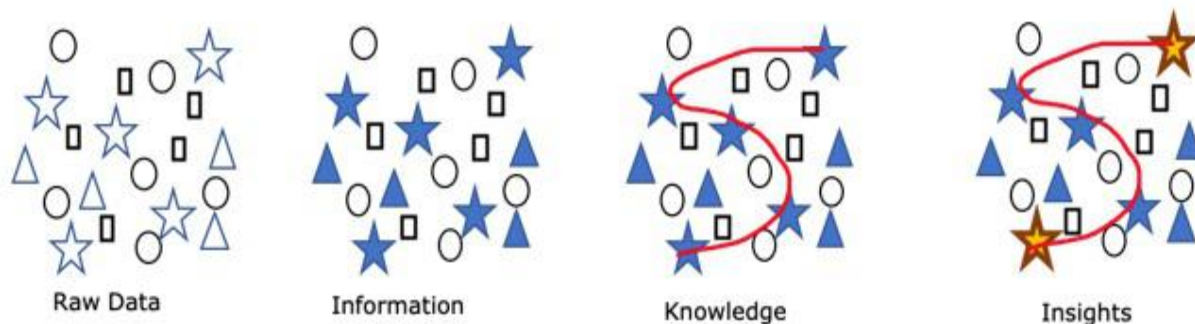
Data in itself is not useful till we clean it, arrange it, analyse it and draw insights from it. We can visualise the transition in (Figure 1-1). Raw data is converted to information when we are able to find distinctions in it. When we relate the terms and *"connect the dots"*, the same piece of information becomes knowledge. Insight is the stage when we are able to find the major centres and significant points. An insight has to be actionable, succinct and direct. For example, if a customer retention team of a telecom operator is told that customers who do not make a call for 9 days have 30% more chances of churn than those who do use, it will be an useful insight on which they can work and try to resolve. Similarly, if a line technician in a manufacturing plant is informed that using Mould ABC results in 60% more defects if used with Mould PQR, they will refrain from using this combination. An insight is quite useful for a business team and hence they can take corrective measures.

We now know what is data. Let us study various types of data and their attributes and go deeper in data.

1.1.2 Various types of Data

Data is generated across all the transactions we make, be it online mode or offline medium, as we discussed at the start of the section. We can broadly classify the data as shown in (Figure 1-2) below:

Figure 1-2 Data can be divided into quantitative and qualitative categories, which are further sub-classified
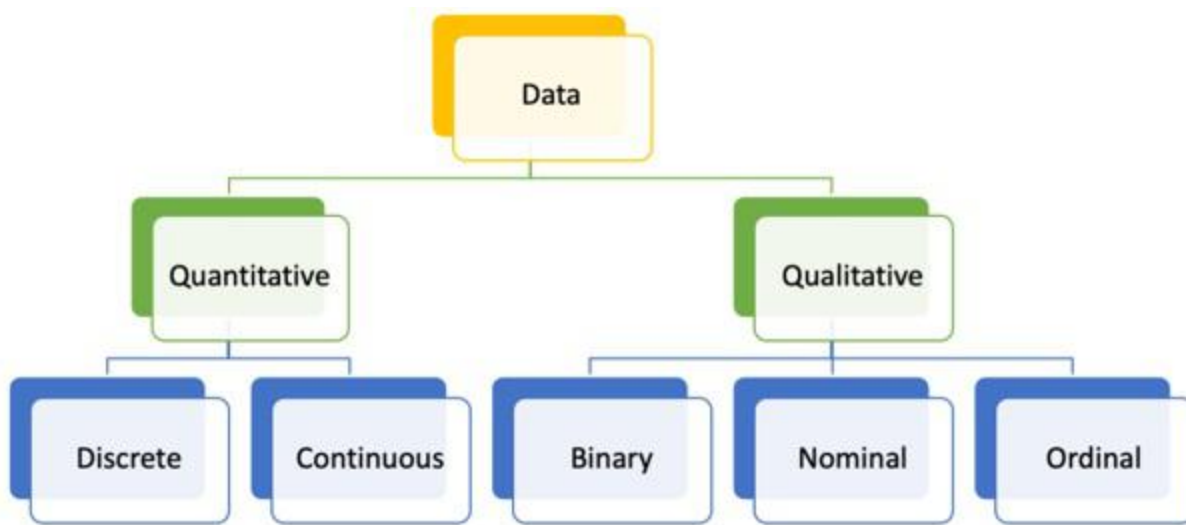
1. **Qualitative Data** is the data type which cannot be measured or weighed for example, taste, colour, odour, fitness, name etc. They can only be observed subjectively. Formally put, when we categorize something or make a classification for it, the data generates is qualitative in nature. For example, colours in a rainbow, cities in a country, quality of a product, gender etc. They are also called as *categorical* variables. Qualitative data can be further sub-categorised in binary, nominal and ordinal data sets.
2. **Binary** data, as the name suggests, has only two classes which are mutually exclusive to each other. For example, Yes/No, dry/wet, hard/soft, good/bad, true/false etc.
3. **Nominal** data can be described as the type of data which though is categorised but does not have any sequence or order in it. For example, distinct languages spoken in a country, colours in a rainbow, types of services available to a customer, cities in a country etc.
4. **Ordinal** data is similar to nominal data except we can order it in a sequence. For example, fast/medium/slow, positive/neutral/negative etc.
5. **Quantitative data:** All the types of data points which can be measured, weighed, scaled, recorded etc. are quantitative. For example, height, revenue, number of customers, demand quantity, area, volume etc. They are the most common form of data and allow mathematical and statistical operations on them too. Quantitative data is further sub-

categorised as discrete and continuous:

6. **Discrete** data are precise, to-the-point and integers. For example, number of passengers in a plane or population of a city cannot be in decimals.
7. **Continuous** data points can take any value, usually in a range. For example, height can take decimal values or price of a product can need not be an integer.

Any data point generally will fall in the above discussed classes. These classes are based on the variable and it's type. There is one more logical grouping which can be done using source and usage, which makes a lot of sense while solving business problems. This grouping allows us to designs solutions customized to the data type.

Depending on the source and usages, we can also think of data in two broad classes: unstructured data and structured data.

1. **Structured data**: dataset which can be represented in a row column structure easily is a structured dataset. For example, transactions made by 5 customers in a retail store can be stored as shown in table below (Table 1-1):

Table 1-1 An example of structured dataset having attributes like amount, data, city, items etc.

Customer ID	Txn Date	Amount ($)	# of items	Payment Mode	City
1001	01-June-2020	100	5	Cash	New Delhi
1002	02-June-2020	101	6	Card	New York

1003	03-June-2020	102	7	Card	London
1004	04-June-2020	103	8	Cash	Dublin
1005	05-June-2020	104	9	Cash	Tokyo

In the table above, for each unique customer ID we have the transaction date, amount spent in $, number of items purchased, the model of payment and the city in which the transaction has been made. Such data type can be extended to employee details, call records, banking transactions etc.

Note:

Most of the data used in analysis and model building is structured. Structured data is easier to store, analyse and visualise in the form of graphs and charts.

We have lot of algorithms and techniques catering to structured data – in normal real-world language we refer to structured data primarily.

2. **Unstructured data**: Unstructured data can be text, audio, image or a video. The examples of unstructured data and their respective sources are given in (Figure 1-3) below, where we explain the primary types of unstructured data: text, images, audio and video along with their examples:

Figure 1-3 Unstructured data along with its various types and examples. This data is usually complex to analyze and generally requires deep learning-based algorithms

To be noted is, our computers and processors understand only binary numbers. So these unstructured data points still need to be represented as numbers so that we can perform mathematical and statistical calculations on them. For example, an image is made up of pixels. If it is a coloured image, each pixel will have RGB (red, green, blue) values and each of the RGB can take a value from (0-255). Hence we will be able to represent an image as a matrix on which further mathematical calculations can be made. Similarly, text, audio and videos can be represented too.

Note:

Mostly deep learning based solutions like CNN, RNN are used for unstructured data. We are going to work on text and image data at a later stage in the book

The representation of unstructured data can be understood as shown below in an example in (Figure 1-4). We have shown a picture of a vacuum cleaner. A portion of the image if represented as a matrix, will look like this. It is only for illustration purpose and not the actual values.

Figure 1-4 An image which is an example of unstructured data can be represented as a matrix to analyze

10	11	11	10	12
5	11	15	5	11
6	14	15	10	10
8	12	20	11	19
10	11	10	15	11

Similarly, we can have the representations of text or audio or video data. Owing to the size and large number of dimensions, unstructured data is complex to process and model and hence mostly deep learning based models serve the purpose. These deep learning models form the background of artificial intelligence based solutions.

These are the broad types of data. We can have more categories like ratios or scales which can be used to define the relationship of one variable with other. All of these data points (whether structured or unstructured) are defined by the way they are generated in real life.

These all data points have to be captured, stored and managed. There are quite a few tools available for managing data which we will be discussing in due course. But before that let us examine one of the most crucial but often less talked subject – *data quality*.

1.1.3 Data quality

"Garbage in, garbage out" – this principle summarises the importance of a good quality data. If the data is not clean, usable, correct and related, we will not be able to solve the business problem at hand. But what is the meaning of "good quality". We have shown the major components of data quality in

(Figure 1-5) below, let's explore them one by one.

Figure 1-5 Data quality is of paramount importance, attributes of a good quality data are shown

The major attributes of good quality data are:

1. **Completeness**: we would expect our dataset to be proper and not missing any values. For example, if we are working on sales data for a year, a good data will have all values for all the 12 months. Then it will be a complete data source. Completeness of a dataset ensures that we are not missing on an important variable or data point.
2. **Valid**: validity of data is it's conformance to the properties, characteristics and variations that are present and being analysed in our use case. Validity indicates if the observation and measurement we have captured is reliable and valid. For example, if the scope of study is for 2015-2019, then using 2014 data will be invalid.
3. **Accurate**: Accuracy is an attribute focussing on the correctness of data. If we have inaccurate data, we will generate inaccurate insights and actions will be faulty. It is a good practice to start the project by generating KPI (key performance indicators) and comparing them with the numbers reported by the business to check the authenticity of the

data available to us.

4. **Representative**: It is one of the most important attribute of the data. And often most undermined too. Representation of a data means, if the data in use truly captures the business need and is not biased. If the dataset is biased or is not representative enough, the model generated will not be able to make predictions on the new and unseen data and the entire efforts will go down the drain.

5. **Available:** Non-availability of data is a challenge we face quite a lot. Data might not be available for the business problem and then we face a dilemma to continue the use case. Sometimes we face operational challenges and do not have access to the database or face permission issues or data might not be available at all for a particular variable since it is not captured. In such cases, we have to work with the data available to us and use surrogate variables. For example, imagine we are working on a demand generation problem. We want to predict how many customers can be expected during the upcoming sale season to a particular store. But we do not historical records of number of customers visiting for few months. We can then use revenue as a surrogate field and synthesize the missing data points for us.

6. **Consistent:** Here we check if the data points are consistent across systems and interfaces. It should not be the case that one system is reporting a different revenue figure while another system is showing a completely different value. When faced with such an issue, we generate the respective KPIs as per the data available to us and seek guidance from the business team.

7. **Timeliness**: It simple means that do we have all the data which is required at this point of time. If the data set is not available now but might become available in the future, then it might be prudent that we wait till then.

8. **Integrity**: The data tables and variables we have are interlinked and interrelated to each other. For example, an employee's details can be spread over multiple tables which are linked to each other using employee ID. Data integrity addresses this requirement and ensures that all such relations between the tables and respective entities is consistent.

A good quality of data is of paramount importance. In pragmatic day-to-day business, often we do *not* get a good quality data. Due to multiple challenges

good clean data, which is accessible, consistent, representative and complete is seldom found on the systems.

Degradation in quality can be due to challenge during data capturing and collection, exporting or loading, transformations done etc. Few of the issues are listed below:

1. We can get integers as names, or special characters like "#$!&" in a few columns, or null values, blanks or NaN (not a number) as some of the values.
2. Duplicates in the records are also one of issues we face.
3. Outliers is one of nuisance we have to deal with quite a lot. For example, let's say that average daily transactions are 1000 for an online retailer. One fine day, due to a server problem there were no transactions done. It is an outlier situation. Or one fine day, the number of transactions were 1,000,000. It is again an example of outlier.
4. Then there are seasonal variations, movements with respect to time of the day and days of the week – all of them should be representative enough in the dataset.
5. Inconsistencies in the date format leads to multiple challenges while we try to merge multiple data sources. Source 1 might be using DD/MM/YYYY while another might be using MM/DD/YYYY. It is to be taken care off during the data loading step itself.

All these aberrations and quality issues have to be addressed and cleaned thoroughly. We will be solving these data issues throughout the book and sharing the best practices to be followed.

Note:

The quality of your raw data and rigour shown during the cleaning process - defines the impact of your final analysis and the maturity of your solution.

We have now defined the major attributes of a data. We will now study the broad process and techniques used for data engineering and management.

1.1.4 Data engineering and management

A strong data engineering process and mature data management practice is a pre-requisite for a success machine learning model solution. Refer to the (Figure 1-6) below where the end-to-end journey of data is described – right from the process of data capturing, data pipeline, data loading to the point it is ready for analysis.

Figure 1-6 Data engineering paves the way for data analysis. It involves data loading, transformation, enrichment, cleaning and preparation etc. which leads to creation of data ready for analysis
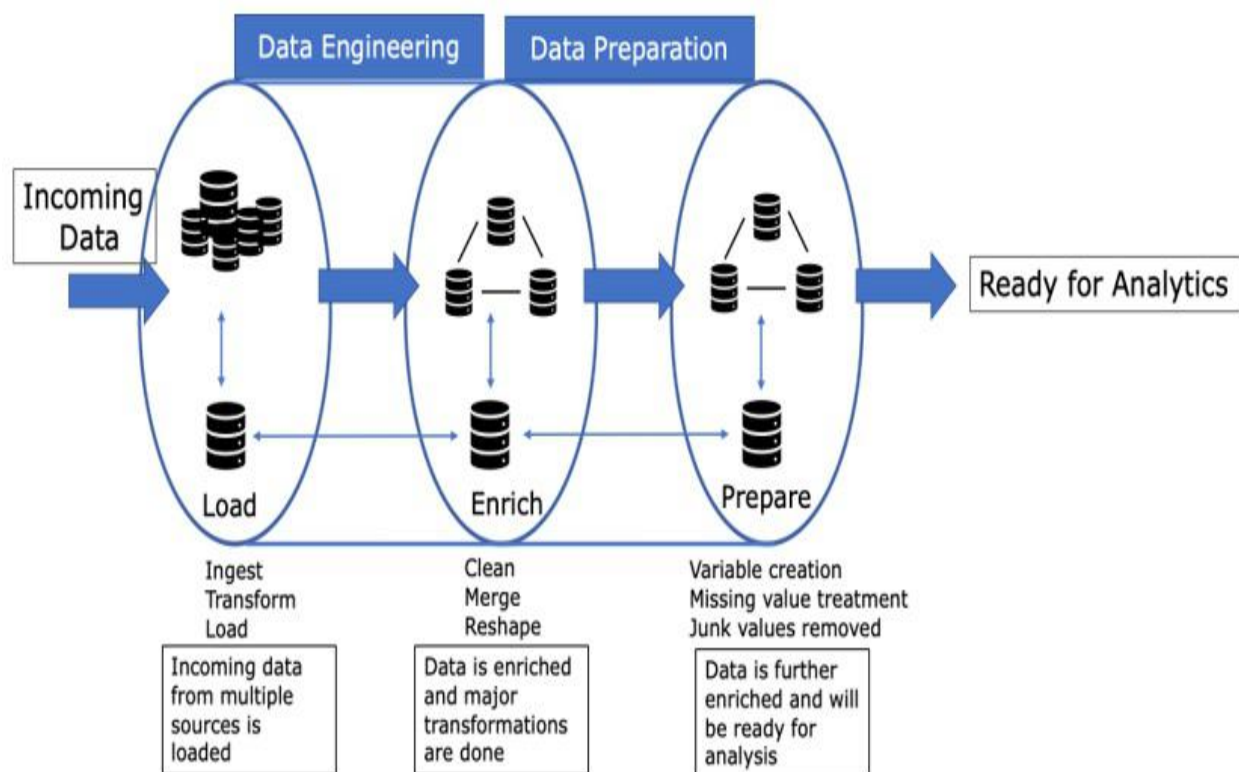
In the data engineering step, data is cleansed, conformed, reshaped, transformed and ingested. Generally we have a server where the final data is hosted and is ready for access,. The most used process is creation of ETL (export, transform, load) processed. Then we make the data ready for analysis. We create new variables, treat null values, enrich the data with methods and then we finally proceed to the analysis/model building stage.

Tip:

It will be a good idea to understand the basics of data engineering to

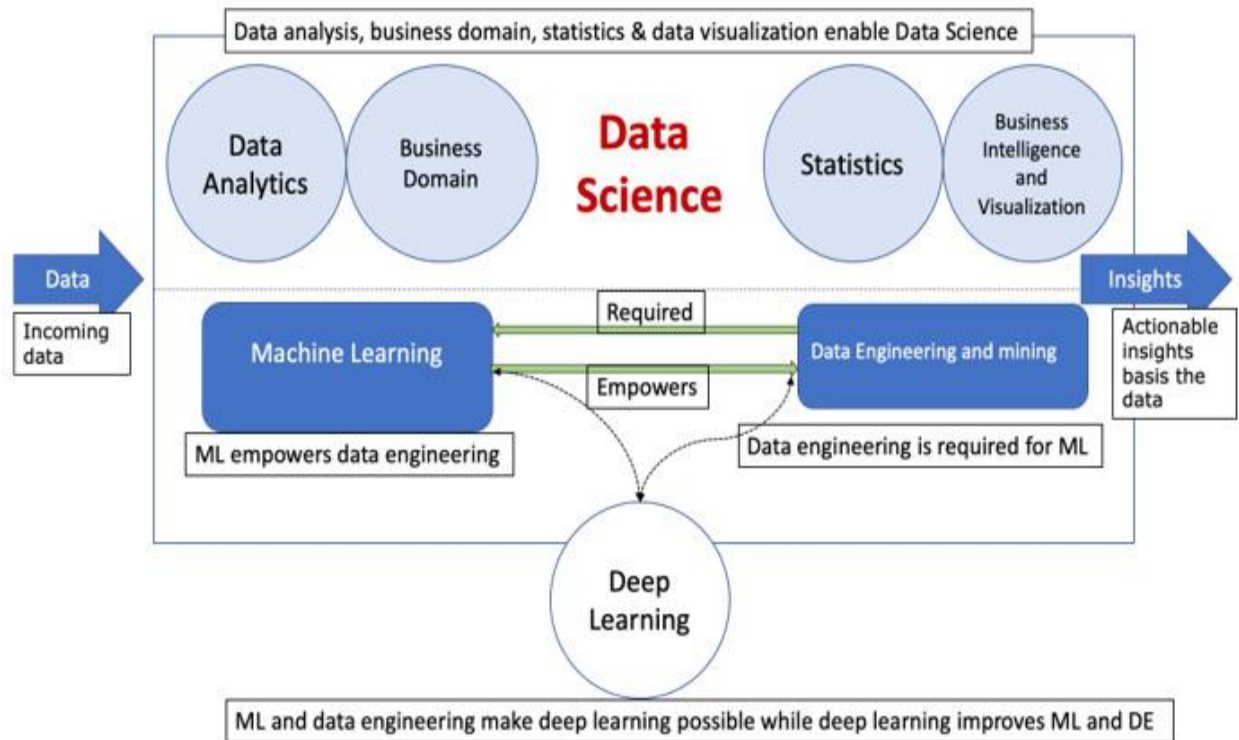complement the knowledge about data science as both go hand-in-hand

We have thus studied what is data and what are the qualities of good data to use. The data is used for analysis, modelling, visualization, dashboards and insight generation. Many times, we find that terms like data analysis, data science, machine learning, data mining, artificial intelligence, business intelligence, big data etc. are used quite interchangeably in the business. It will be a good idea to clarify them – which is the topic of the next section. There are plenty of tools available for each respective function, which we are discussing. And we will also understand the role of software engineering in this entire journey.

1.2 Data Analysis, Machine Learning, Artificial Intelligence and Business Intelligence

Data and its importance have opened new avenues and created a lot of job opportunities in the market. At the same time, machine learning and artificial intelligence being a relatively newer field, there are not much standardisation and differentiation in the scope of work. It has resulted in not-so-clear definitions and demarcation of these fields. We are examining these fields – where they overlap, where they differ and how one empowers the another. It can be visualised by the means of a diagram below in (Figure 1-7):

Each of the function empowers each other and complements each other.

Figure 1-7 How the various fields are interlinked with each other and how they are dependent on each other

Data analysis, business domain, statistics & data visualization enable Data Science

Data Analytics · Business Domain · Data Science · Statistics · Business Intelligence and Visualization

Data / Incoming data → Machine Learning — Required / Empowers — Data Engineering and mining → Insights / Actionable insights basis the data

ML empowers data engineering · Data engineering is required for ML

Deep Learning

ML and data engineering make deep learning possible while deep learning improves ML and DE

Data mining and data engineering starts all of it by providing the data required for analysis. It also exports, transforms, cleans, loads so that it can be consumed by all of the respective functions. Business intelligence and visualisations use this data to generate reports and dashboards. Data analytics generates insights and trends using data. Data Science stands on the pillars of data analysis, statistics, business intelligence, statistics, data visualization, machine learning and data mining. Machine learning creates statistical and mathematical models. And artificial intelligence further pushes the capabilities.

Machine learning uses traditional coding. The coding is done in traditional languages and hence all the logics and rules of computer science and software engineering are valid in machine learning too. Machine learning helps us in making sense of the data which we are otherwise not able to comprehend. And in that aspect, it is a fantastic solution. It is able to relate to the historical trends. The most fascinating advantage with machine learning is its ability to work on very complex and high dimensional data points like video, audio, image, text or complex datasets generated by sensors. It allows us to think beyond the obvious. Now artificial intelligence is able to achieve the feats which were previously thought not possible at all. Like self-driving

cars, chat-bots conversing like humans, speech-to-text conversion and translation to other language of choice, automated grading of essays, photo captioning etc.

We are now clear how the various fields are different from each other yet interlinked and how machine learning is different from traditional software engineering. It is the foundation of our under topic of discussion, which goes deeper into machine learning and its various components, different types of machine learning algorithms and their respective use cases.

1.3 Nuts and bolts of Machine Learning

Consider this. If a child has to be taught how to open a door knob, we show her the exact steps quite a few times. The child tries to open, fails. Tries again and fails again. But in each subsequent try, the child is improvising the approach. And after sometime, the child is able to open the door knob. Or when we try to learn driving, we make mistakes, we learn from them and we improve. *Machine Learning* works similarly - wherein the statistical algorithm looks at the historical data and find patterns and insights. The algorithm uncovers relationships and anomalies, trends and deviations, similarities and differences – and then shares back actionable results to us.

Formally put, machine learning can be called as a branch or a study of computer algorithms which works on historical data to generate insights and helps in making data-driven decisions. The algorithms are based on statistical and mathematical foundation and hence have a sound logical explanation. Machine Learning algorithms require coding which can be done in any of the language and tools available viz. Python, R, SPSS, SAS, MATLAB, Weka, Julia, Java etc. It also requires domain understanding of the business.

Note:

Languages are only means to an end. All the languages generate similar results for a machine learning algorithm even if used across different languages.

So whenever you are doing some online shopping for a dress and the website

recommends you accessories which go along with it, or you are booking an air-ticket and the travel operator shows you a customized deal as per your needs and plan – machine learning is in the background. It has learnt your preferences and compared with your historical trends. It is also looking for similarities you have with other customers who behave almost the same. And based on all of those analysis, the algorithm is making an intelligent recommendation to you. Quite fascinating, right!

Many times we ask this question, why do we require machine learning and why it surpasses human intelligence? The reason is, we humans can analyse only two or many be three dimensions simultaneously. But a machine learning algorithm can work on 50,60 or may be 100s of dimensions simultaneously. It can work on any type of data, be it structured or unstructured and can help in automation of tasks. And hence it generates patterns and insights quite difficult for a human mind to visualise.

Machine Learning like any other project requires a team of experts who are working closely with each other and complementing each other's skillsets. As shown in the (Figure 1-8) below, a machine learning project requires the following roles:

- **Business stakeholders and Subject Matter Experts (SME):** They define the business problem for the project. They own the solution, have a clear understanding of the ask and have a clear measurable goal in sight. They course correct the team in case of confusions and serve as an expert who have a deep understanding of the business processes and operations. They are marketing managers, product owners, process engineers, quality experts, risk analysts, portfolio leads etc.

Note:

It is imperative that business stakeholders are closely knit in the team from day 1.

- **Operations Team:** This team comprises of the scrum master, project manager, business analysts etc. The role of the team can be compared to typical project management team which tracks the progress, maintains the records, reports the day-to-day activities and keep the entire project

on track. They create user stories and act as a bridge between the business team and data team.

Figure 1-8 Team required for a data science project and the respective interactions of them with each other
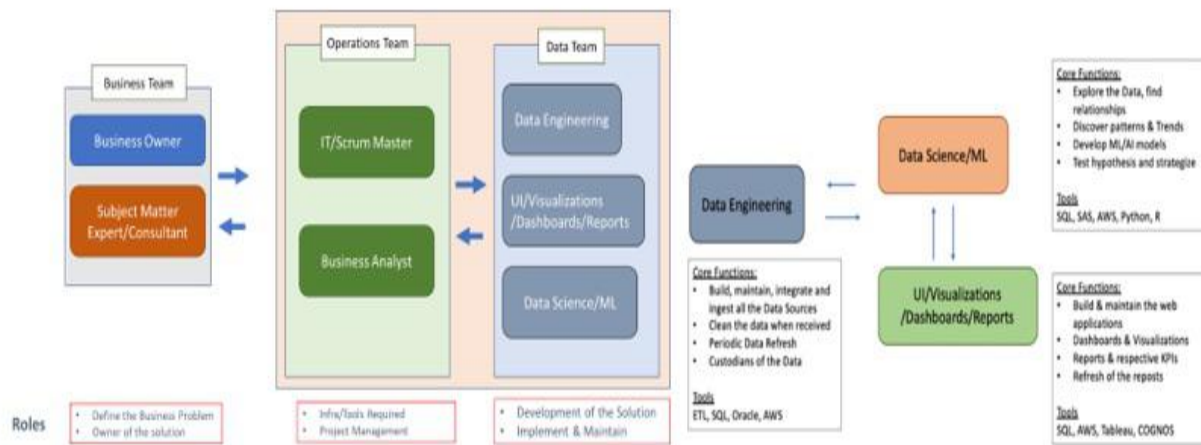

- **Data Team:** The core team which creates the solution, does the coding, generates the output in the form of a model, dashboard, report, insights is the data team. It comprises of three main pillars: Data engineering, UI/Visualization team and the Data Science team. There functions are as follows:
- o Data engineering team is responsible for building, maintaining, integrating and ingesting all the data points. They do a periodic data refresh and act as a prime custodian of data. They use ETL, SQL, AWS, Kafka etc.
- o UI/ Visualisation team builds dashboards, reports, interactive modules and web applications. They use SQL, Tableau. Qlik, Power BI etc.
- o Data Science team is responsible for all the data analysis and model building tasks. They discover patterns and insights, test hypothesis and generate the final output which is to be finally consumed by all. The final output can be a machine learning model which will be used to solve the business problem. In situations where a machine learning model is not possible, the team might generate actionable insights which can be useful for the business. This team requires SQL, Python, R, SAS, SPSS etc. to complete their job.

We have understood the typical team structure for a data science project. We will not examine what are the broad steps in a data science project.

A data science project runs like any other project having deadlines, stages, testing, phases etc. The raw material is the data which passes through various phase to be cleaned, analysed and modelled.

We are shown an illustration of a data science project stages in (Figure 1-9) below.

Figure 1-9 Data science project is like any other project, having stages and deadlines, dependencies and processes
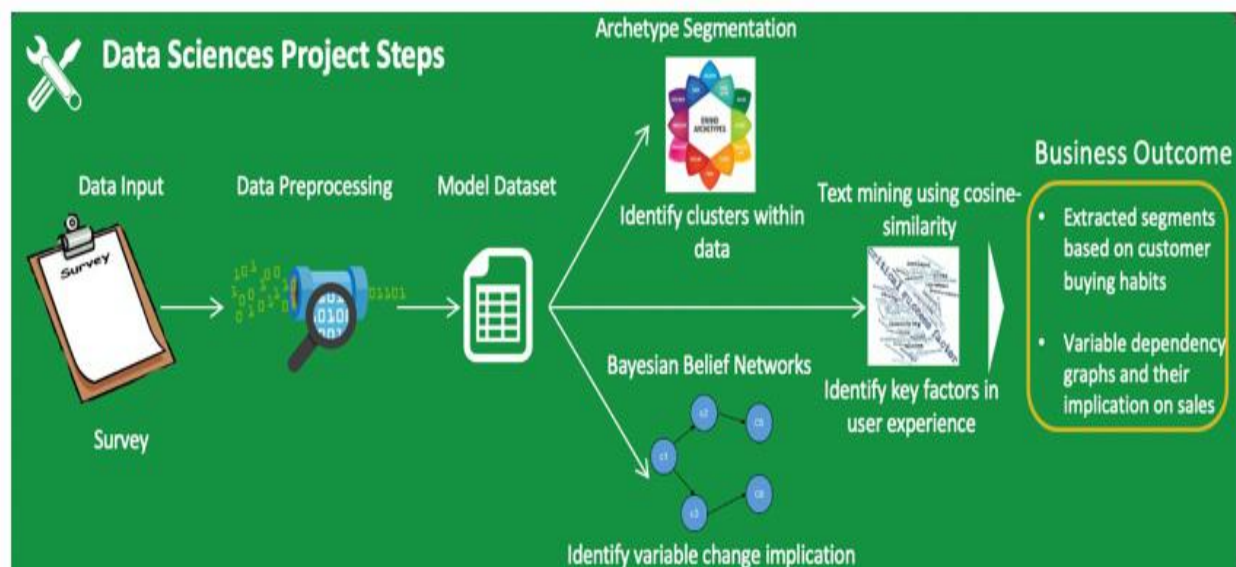
It starts with a business problem definition of the project. Business problem has to be concise, clear, measurable and achievable. The table (Table 1-2) below depicts example of a bad and a good business problem.

Table 1-2 Examples on how to define a business problem to make it clear, concise and measurable

Examples of an ill-defined business problem	Example of a good business problem

Increase the production	Optimize the various cost heads (A,B, C and D) and identify the most optimal combination to decrease the cost by 1.2% in next 6 months
Decrease the cost	
Increase the revenue by 80% in 1 month	From the various factors of defects in the process (X,Y,Z), identify the most significant factors to reduce the defect % by 1.8% in next 3 months
Automate the entire process	

Then we move to the data discovery phase during which we list down all the data sources and host them. All the various datasets like customer's details, purchase histories, social media data, portfolio etc. are identified and accessed. The data tables which are to be used are finalised in this step and most of the time, we create a database for us to work, test and learn.

We go ahead with data pre-processing. It involves cleaning of data like removal of null values, outliers, duplicates, junk values etc. The previous step and this one can take 60-70% of the project time.

We create few reports and generate initial insights during the exploratory data analysis phase. These insights are discussed with the business stakeholders and their guidance is taken for course correction.

The data is now ready for modelling. Quite a few versions of the solution are tested. And depending on the requirements we choose the best version. Mostly parameters like accuracy and statistical measures like precision, recall drive the selection of the model. We will be exploring the process to choose the best model and terms like precision, recall in later chapters of the book.

The final model is chosen and now we are ready for deploying the model in the production environment where it will work on unseen data.

These are the broad steps in a machine learning project. Like any other project, there is a code repository, best practices and coding standards, common error, pitfalls etc. which we are discussing throughout the book.

We will now move to one of the important topics which is types of machine learning algorithms, which we are discussing now.

1.4 Types of Machine Learning algorithms

Machine Learning models are impacting the decision making and follow a statistical approach to solve a business problem. It works on historical data and finds patterns and trends in it. The raw material is the historical data which is analysed and modelled to generate a predictive algorithm. This historical data available and the business problem to be solved allow us to classify the machine learning algorithms in broadly four classes: **supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning** as depicted in the (Figure 1-10) below. We are examining all of the four types in detail now with a focus on unsupervised learning which is the topic of this book.

Figure 1-10 Machine learning algorithms can be classified as supervised learning algorithms, unsupervised learning algorithms, semi-supervised learning algorithms and reinforcement learning algorithms
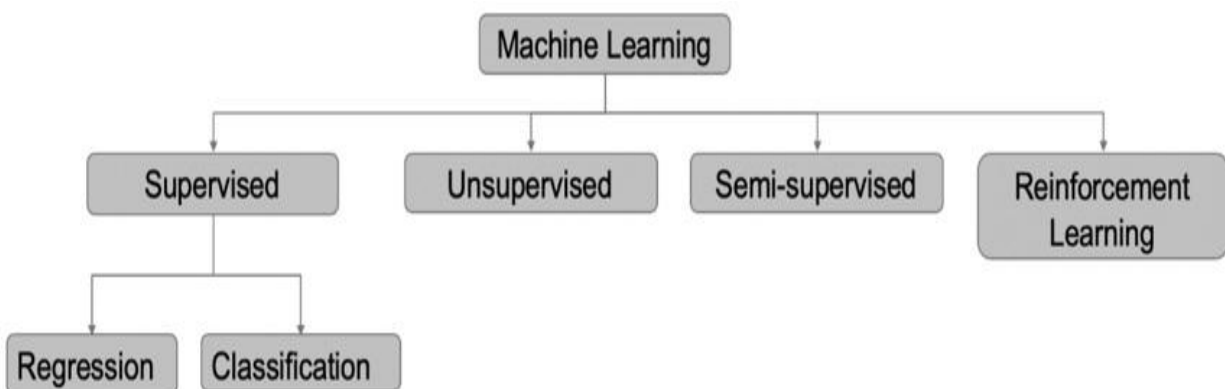
1.4.1 Supervised Learning

As the name suggests, supervised learning algorithm have a "guidance" or "supervision" to direct toward the business goal of predicting for the future.

Formally put, supervised models are statistical models which use both the input data and the desired output to predict for the future. The output is the value which we wish to predict and is referred as the *target variable* and the data used to make that prediction is called as *training data*. Target variable is sometimes referred as the *label*. The various attributes or variables present in the data are called as *independent variables*. Each of the historical data point or a *training example* contains these independent variables and corresponding target variable. Supervised learning algorithms make a prediction for the unseen future data. The accuracy of the solution depends on the training done and patterns learned from the labelled historical data. The example to describe the concept is in the next section.

Supervised learning problems are used in demand prediction, credit card fraud detection, customer churn prediction, premium estimation etc. They are heavily used across retail, telecom, banking and finance, aviation, insurance etc.

Supervised learning algorithms can be further broken into regression algorithms and classification algorithms. We will first work with Regression problems.
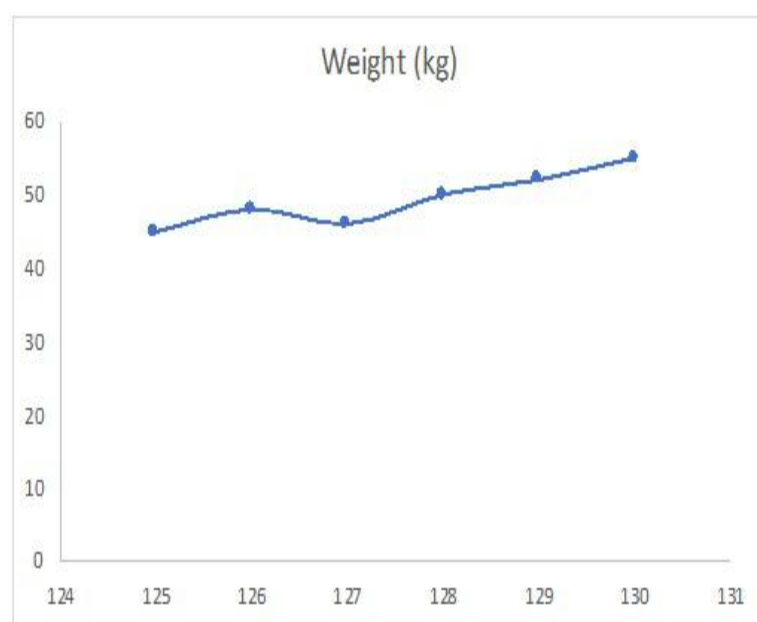
Regression algorithms

Regression algorithms are supervised learning algorithms i.e. they require a target variables which needs to be predicted. These algorithms are used to predict the values of a *continuous* variable. For example, revenue, amount of rainfall, number of transactions, production yield and so on. In supervised classification problems, we predict a categorical variable like whether it will rain (yes/no), whether the credit card transaction is fraud or genuine and so on. This is the main difference between classification and regression problems.

Let us understand regression problem with an example. If we assume that the weight of a person is only dependent on height and not on other parameters like gender, ethnicity, diet etc. In such a case, we want to predict the weight of a person based on the height. The dataset can look like below and the graph plotted for the same data will look like as shown below in (Figure 1-11).

A regression model will be able to find the inherent patterns in the data and fit a mathematical equation describing the relationship. It can then take height as an input and predict the weight. Here height is the independent variable and weight is the dependent variable or the target variable or the label we want to predict.

Figure 1-11 Data and plot of relationship between height and weight which is used for regression problem

Height (cm)	Weight (kg)
125	45
126	48
127	46
128	50
129	52
130	55

There are quite a few algorithms available for regression problems, the major ones are listed below:

1. Linear regression
2. Decision tree
3. Random forest

4. K-nearest neighbour
5. Boosting algorithms
6. Neural network

We can use any of the algorithms to solve this problem. We will explore more by using a **linear regression** to solve this problem.

Linear regression algorithm models the relationship between dependent and target variables by assuming a linear relationship exists between them. The linear regression algorithm would result in a mathematical equation for the problem as shown in the equation (1-1)

Weight = β_0 * height + β_1 (Equation 1-1)

Generally put, a linear regression is used to fit a mathematical equation depicting the relationship between dependent and independent variables as shown in (1-2).

$Y = \beta 0 + \beta 1\ x1 + \beta 2 x2 + \ldots.+\varepsilon$ (Equation 1-2)

here Y is the target variable which we want to predict

x_1 is the first independent variable

x_2 is the second independent variable

ε is the error term in the equation

β_0 is the intercept of the equation

A simple visualization for a linear regression problem is shown in (Figure 1-12). Here, we have the x and Y variables where x is independent variable and Y is the target variable. The objective of the linear regression problem is to find the *line of best fit* which is able to explain the randomness present in the data.

Figure 1-12 Raw data which needs to be modelled is on the left. Using regression, a line of best fit

is identified
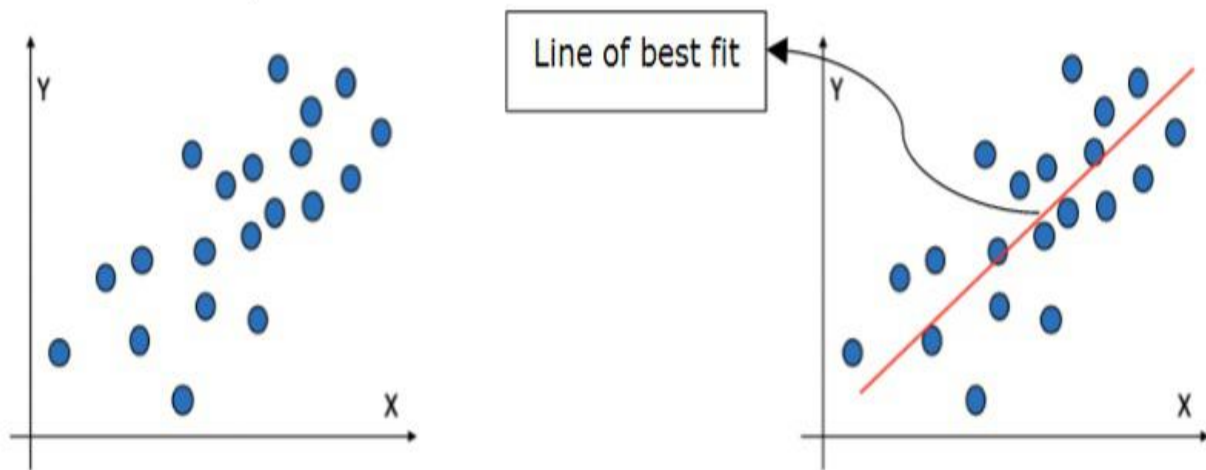
This equation is used to make predictions for the unseen data. There are variations in linear regression too like simple linear regression, multiple linear regression, non-linear regression etc. Depending on the data at hand, we choose the correct algorithm. A complex dataset requires might require a non-linear relationship between the various variables.

The next regression algorithm we are discussing is **Tree based solutions**. For tree based algorithms like decision tree, random forest etc. the algorithm will start from the top and then like an "if-else" block will split iteratively to create nodes and sub-nodes till we reach a terminal node. It can be understood better by the means of (Figure 1-13). In the decision tree diagram, we start from the top with the root node and then splitting is done till we reach the end point which is terminal node.

Figure 1-13 Decision tree has a root node and after splitting we get a decision node and terminal node is the final node which cannot be split further
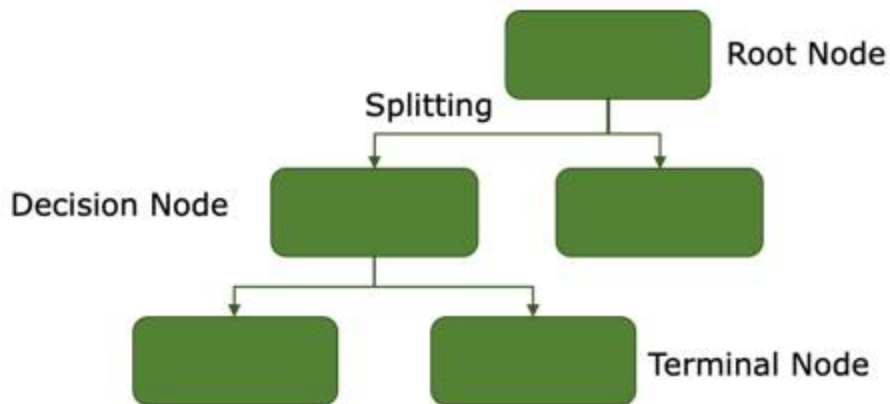
Decision tree is very easy to comprehend, implement and fast to train. Their usability lies in the fact that they are intuitive enough to understand by everyone.

There are other famous regression algorithms like k-nearest neighbour, gradient boosting, deep learning based solutions. Based on the business problem and respective accuracies we prefer one regression algorithm over another.

To understand the impact of regression use cases, there are a few business relevant use cases which are implemented in the industry:

1. An airport operations team is doing an assessment of the staffing requirement and want to estimate the number of passenger traffic expected. The estimate will help the team to prepare a plan for the future. It will result in optimization of resources required. Regression algorithms can be of help in predicting the passengers.
2. A retailer wants to understand what is the expected demand for the upcoming sales season so that the inventory can be planned for various goods. It will result in cost saving and avoiding stock-outs. Regression algorithms can help in such a planning.
3. A manufacturing plant wishes to improve the yield from the existing usage of various moulds and raw material. The regression solutions can suggest the best combination of moulds and predict the expected yield too.
4. A bank offers credit cards to its customers. But what should be the credit limit offered to a new customers is to be resolved. Based on the

attributes of customer like age, occupation, income, previous transaction history – regression algorithms can help in suggesting credit limit at a customer level.

5. An insurance company wishes to come up with a premium table for its customers using historical claims. The risk can be assesses based on the historical data around driver details, car information etc. Regression can surely help with such problems.

Regression problems form the basics of supervised learning problems and are quite heavily used in the industry. Along with classification algorithms, they serve as a go-to solution for most of the predictive problems which we are discussing now.

Classification algorithms

Simply put, classification algorithms are used to predict the values of a categorical variable which is the dependent variable. This target variable can be binary (Yes/No, good/bad, fraud/genuine, pass/fail etc.) or multi-class (positive/negative/neutral, Yes/No/Don't know etc.). Classification algorithms will ascertain whether the target event will happen or not by generating probability score for the target variable.

After the model has been trained on historical data, a classification algorithm will generate a probability score for the unseen dataset which can be used to make the final decision. Depending on the number of classes present in the target variable, our business decision will vary.

Let's have a look at a use case for classification problems.

Consider this. A telecom operator is facing issue with its decreasing subscriber base. The number of existing subscribers is shrinking and the telecom operator would like to arrest this churn of subscribers. And for this purpose a machine learning model is envisioned.

In this case, the historical data or the training data available for model building can look like the table below in (Table 1-3). These data points are only for illustration purpose and not exhaustive. There can be many other

significant variables available.

Table 1-3 Example of a structured dataset for a telecom operator showing multiple data attributes

ID	Revenue($)	Duration of service (years)	Avg. Cost	Monthly usage (days)	Churned (Y/N)
1001	100	1.1	0.10	10	Y
1002	200	4.1	0.09	25	N
1003	300	5.2	0.05	28	N
1004	200	0.9	0.25	11	Y
1005	100	0.5	0.45	12	Y

In the above example, the dataset comprises of the past usage data of subscribers. The last column (Churned) depicts if that subscriber churned out of system or now. Like subscriber # 1001 churned while 1002 did not. Hence the business problem is, we want to build a machine learning model based on this historical data and predict if a new unseen customer will churn or not.

Here, "churned" status (Yes/No) is the target variable. It is also referred as dependent variable. The other attributes like revenue, duration, average cost, monthly usage etc. are independent variables which are used to create the machine learning model. The historical data is called as the training data. Post the training of the model, the trained supervised learning model will generate

prediction probabilities for a new customer.

There are quite a few algorithms available for classification problems, the major ones are listed below:

1. Logistic Regression
2. Decision tree
3. Random forest
4. K-nearest neighbour
5. Naïve Bayes
6. Support Vector Machine
7. Boosting algorithms
8. Neural network

We will discuss one of most popular classification algorithm called **logistic regression**. Logistic regression uses a logit function to model the classification problem. If we are solving for a binary classification problem, it will be binary logistic regression else multiple logistic regression.

Similar to linear regression, logistic regression also fits an equation, albeit it uses a sigmoid function to generate the probability score for the event to happen or not.

A sigmoid function is a mathematical function which has a characteristic "S" shaped curve or a sigmoid curve. The mathematical equation of a sigmoid function is:

$S(x) = 1/(1 + e^x)$ which can be rewritten as $S(x) = e^x/(e^x + 1)$

The logistic regression uses sigmoid function. The equation used in logistic regression problem is:

$\log (p/1\text{-}p) = \beta_0 + \beta_1 x_1$

where p: probability for the event to happen

β_0 : intercept term

β_1 : coefficient for the independent variable x_1

log(p/1-p) is called the logit and (p/1-p) is the odds

As depicted in the (Figure 1-14) below, if we try to fit a linear regression equation for the probability function, it will not do a good job. We want to obtain the probability scores (i.e. a value between 0 and 1). The linear regression will not only return values between 0 and 1 but also probability scores which are greater than 1 or less than 0. Hence, we have a sigmoid function of the right which generates probability scores for us between 0 and 1 only.

Figure 1-14 Linear regression model will not be able to do justice (left) hence we have logistic regression for classification. Linear regression can generate probability scores more than 1 or less than 0 too, which is mathematically incorrect. Whereas, sigmoid function generates probability scores between 0 and 1 only.
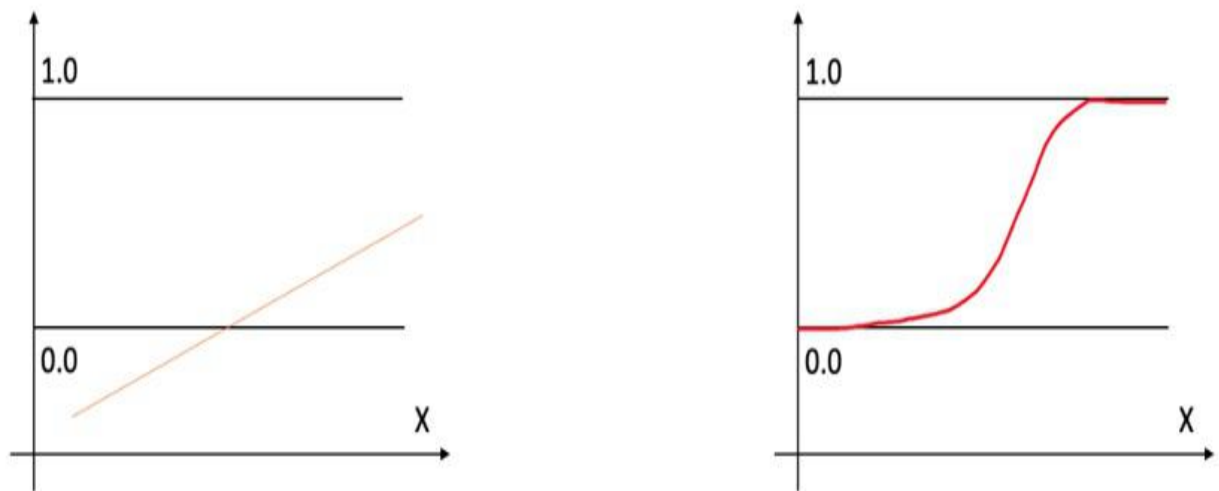
Logistic regression algorithm is one of the most widely used technique for classification problems. It is easy to train and deploy and is often the benchmark algorithm whenever we start any supervised classification learning project.

Tree based algorithms like decision tree, random forest can also be used for classification problems. The other algorithms are also used as per the requirements.

We have studied supervised learning algorithms briefly. We will now discuss unsupervised learning algorithms in the next section – the main topic of this book and then move to semi-supervised learning algorithms.

1.4.2 Unsupervised algorithms

Imagine you are given some paper labels like shown in the figure (Figure 1-15) below. The ask is to arrange them using some similarity. Now there are multiple approaches to that problem. You can use colour, or shape or size. Here we do not have any label with us to guide on this arrangement. This is the difference which unsupervised algorithm have.

Figure 1-15 Example of various shapes which can be clubbed together using different parameters
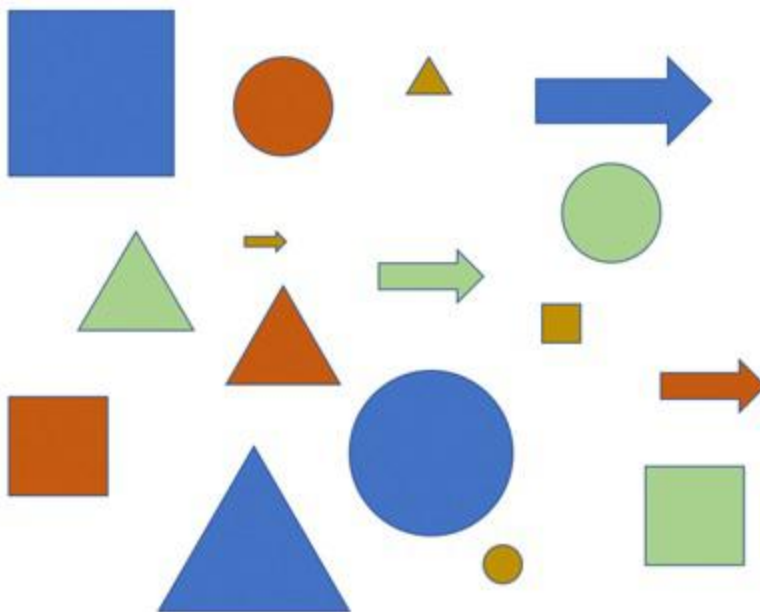

Formally put, unsupervised learning only take the input data and then find patterns in them without referencing to the target variable. An unsupervised learning algorithm hence reacts based on the presence or lack of patterns in the dataset.

Unsupervised learning is hence used for pattern detection, exploring the insights in the dataset and understanding the structure of it, segmentation and anomaly detection.

We can understand unsupervised learning algorithms by the means of (Figure 1-16) below. The figure on the left shows the raw datapoints represented in a vector space diagram. On the right is the clustering done which will be done using unsupervised learning algorithm.

Figure 1-16 Unsupervised learning algorithm find patterns in the data on the left and results in clusters on the right.
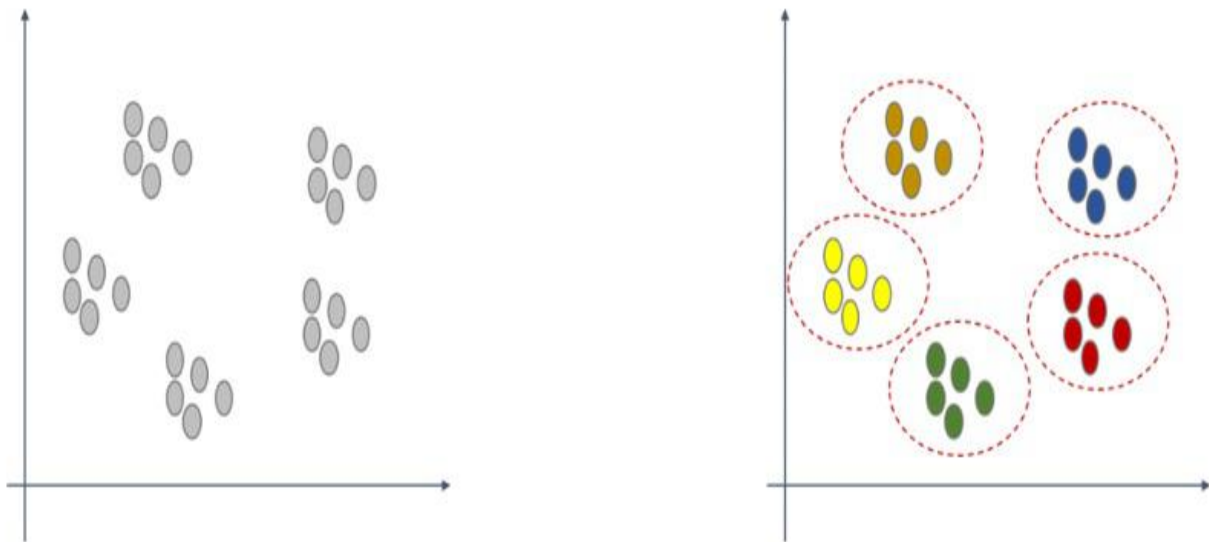
The use cases for unsupervised algorithms are:

1. A retail group wants to understand the customers better. The ask is to improve the customer's stickiness, revenue, number of visits, basket size etc. Customer segmentation using unsupervised learning can be done here. Depending on the customer's attributes like revenue, number of visits, last visit date, age since joining, demographic attributes etc. the segmentation will result in clusters which can be targeted personally. The result will be improved customer experience, increased customer life time value etc.
2. A network provider requires to create an anomaly detection system. The historical data will serve as the anomalies data. The unsupervised learning algorithm will be able to find patterns and the outliers will be given out by the algorithm. The distinguished anomalies will be the ones which need to be addressed.
3. An medical product company wishes to find if there is any underlying

patterns in the images data of their patients. If there are any patterns and factors, those patients can be treated better and may be they require a different kind of approach. Unsupervised learning can help on the images data which will help in addressing the patients better.

4. A digital marketing company wants to understand the "unknowns" in the incoming customer data like social media interactions, page clicks, comments, stars etc.. The understanding will help in improving customer's recommendations and overall purchasing experience.

Unsupervised learning algorithms offer flexibility and performance when it comes to funding the patterns. They are usable for all kinds of data – structured data or text or images or text.

The number of unsupervised learning algorithms are lesser than supervised learning. The major unsupervised learning algorithms are:

1. Clustering algorithms
2. k-means clustering
3. Hierarchical clustering
4. DB Scan clustering
5. Spectral clustering
6. Principal component analysis
7. Singular Value Decomposition
8. Association rules
9. t-SNE (t-distributed stochastic neighbour embedding)
10. Autoencoders

We will be covering all of these algorithms in detail in the coming chapters. We will examine the mathematical concepts, the hidden processes, Python implementation and the best practices throughout the book.

Let us understand by means of a case study.

A retailer wants to develop deeper understanding of its consumer base. And then wants to offer personalised recommendation, promotions, discounts, offers etc. The entire customer dataset has to segmented using attributes like persona, previous purchase, response, external data etc.

For the use case, the steps which are followed in an unsupervised learning project are shown in the (Figure 1-17) below.

Step 1: We start the project by defining the business problem. We wish to understand the customer base better. A customer segmentation approach can be a good solution. We want segments which are distinguishable using mathematical KPIs (key performance indicators).

Figure 1-17 Steps in an unsupervised learning algorithm from data sources to the final solution ready for deployment
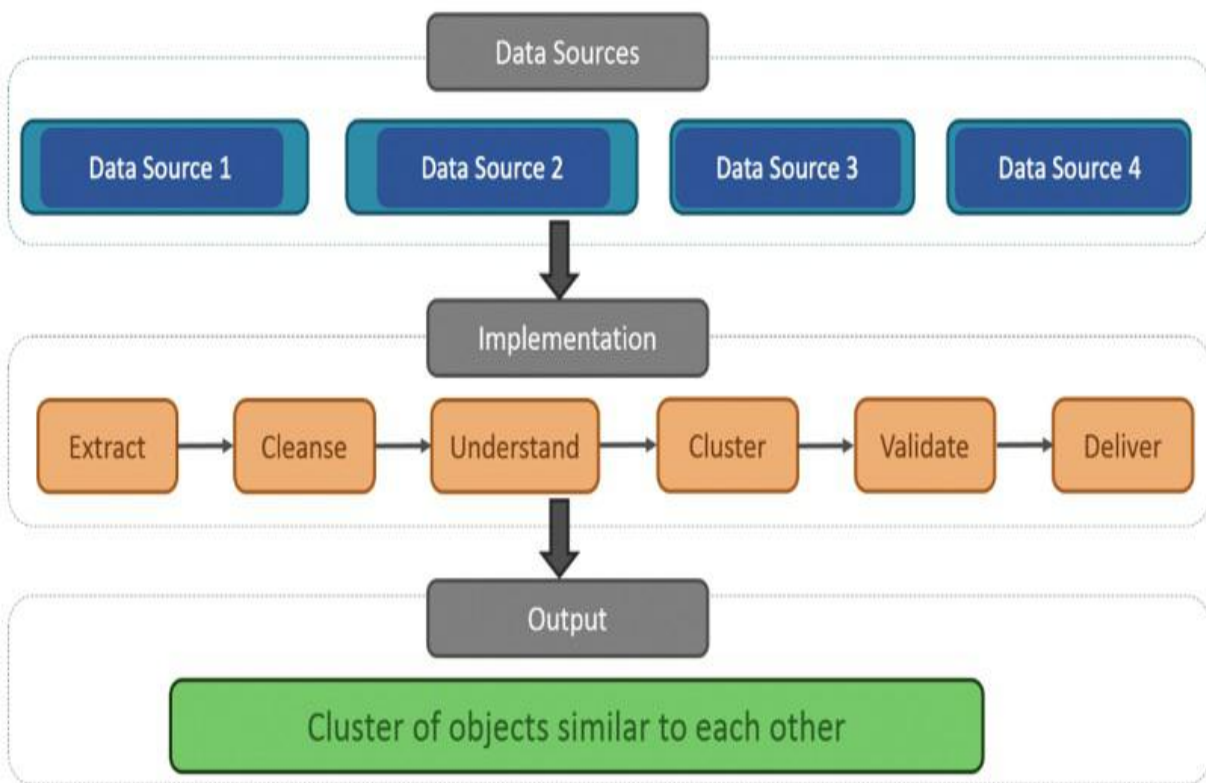

Step 2: This is the data discovery phase. All the various datasets like customer's details, purchase histories, social media data, portfolio etc. are identified and accessed. The data tables to be used are finalised in this step. Then all the data tables are generally loaded to a common database, which we will use to analyse, test and learn.

Step 3: Now we have access to the data. The next step is to clean it and make is usable.

We will treat all the null values, NAN, junk values, duplicates etc.

Step 4: Once the data is clean and ready to be used, we will perform an exploratory data analysis of it. Usually during exploratory analysis, we identify patterns, cyclicity, aberrations, max-min range, standard deviation etc. The outputs of EDA stage will be insights and understandings. We will also generate few graphs and charts as shown below in (Figure 1-17):

Figure 1-17 Examples of the graphs and charts from the exploratory data analysis of the data
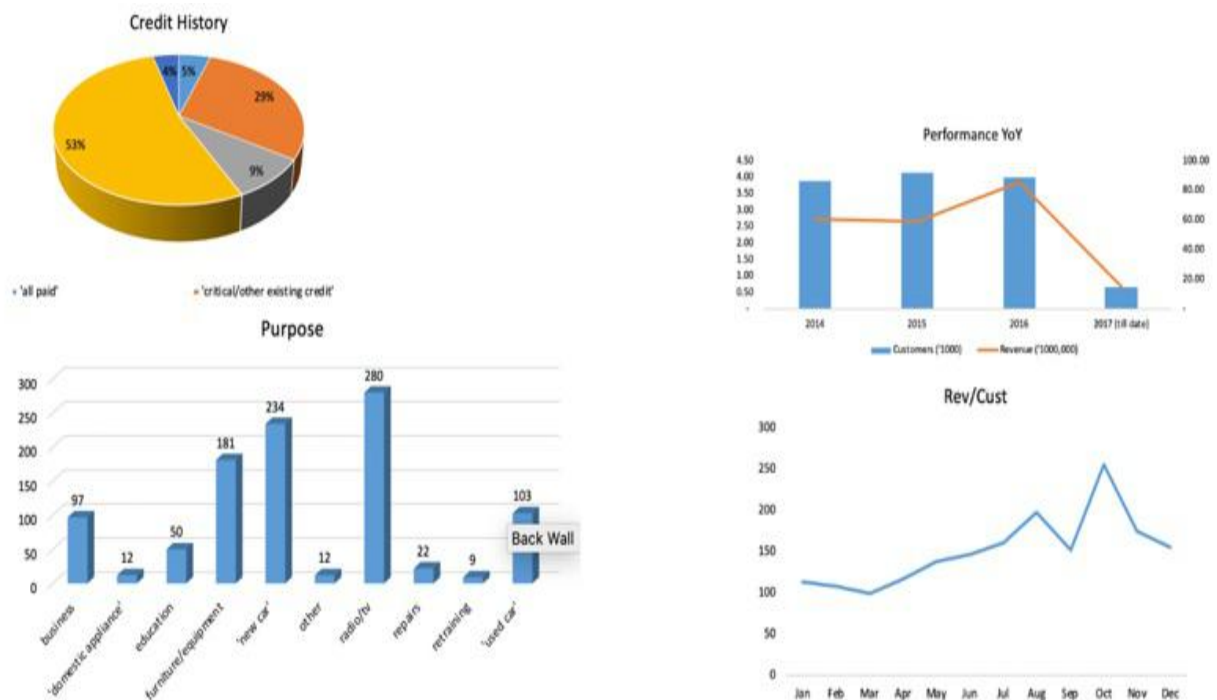
Step 6: We will begin with the unsupervised approach now. We want to implement clustering methods and hence we can try a few clustering methods like k-means, hierarchical clustering etc. The clustering algorithms will result in homogeneous segments of customers based on their various attributes.

In the case study, we will be working on last 2-3 years of data which is the training data. Since we are using an unsupervised approach there is no target variable over here. The algorithm will merge the customer segments which behave alike using their transactional patterns, their demographic patterns and their purchase preferences. It will look like the figure below in (Figure 1-18):

Figure 1-18 Output of the clustering algorithm where we can segment customers using various attributes
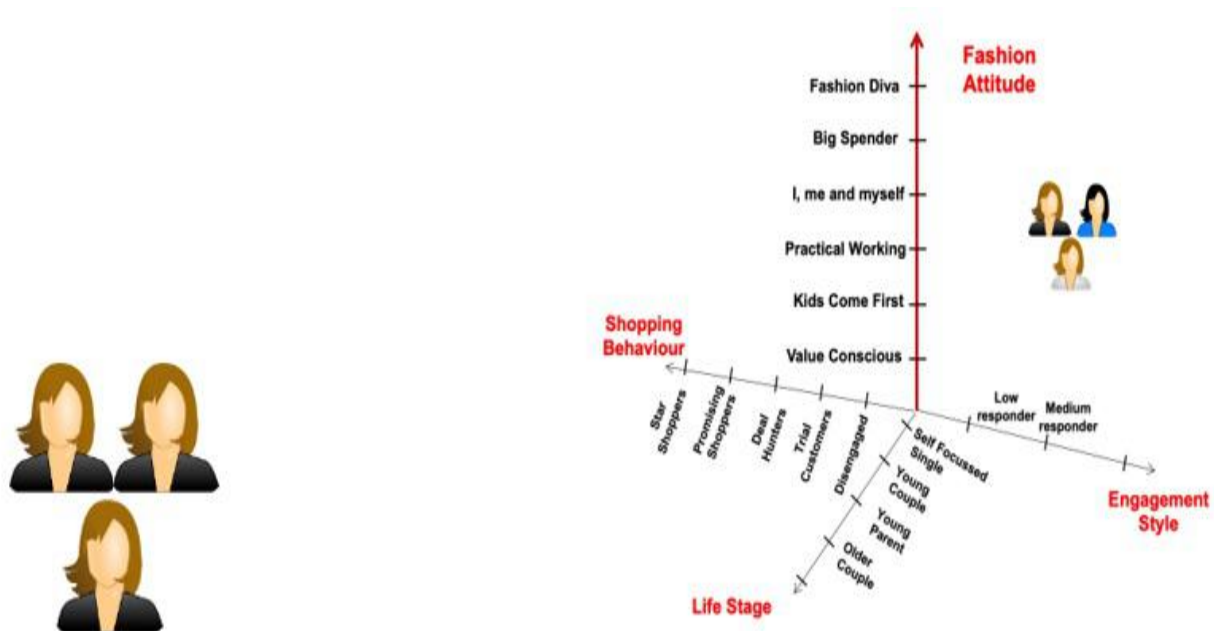
Step 7: we will now check how the various algorithms have performed or in other words we will compare the accuracy of each algorithm. The final clustering algorithm chosen will result in homogeneous segments of customers which can be targeted and offered customized offers.

Step 8: we will discuss the results with the business stakeholders. Sometimes, utilizing the business acumen we merge or break a few segments.

Step 9: Deploy the solution in production environment and we are ready to work on new unseen datasets.

These are the broad steps in an unsupervised problem. The algorithm creation and selection is a tedious task. We will be studying it in details in the book.

So far we have discussed supervised and unsupervised problem. Next we move to semi-supervised algorithms which lie at juxtaposition of supervised and unsupervised algorithms.

1.4.3 Semi-supervised algorithms

Semi-supervised learning is a middle-path of the two approaches. The primary reason of a semi-supervised approach is lack of availability of a complete *labelled* dataset for training.

Formally put, semi-supervised approach uses both supervised and unsupervised approaches – supervised to classify the data points and unsupervised to group them together.

In semi-supervised learning, we train initially on less number of labelled datapoints available using a supervised algorithm. And then we use it to label or *pseudo-label* new datapoints. The two dataset (labelled and pseudo-labelled) are combined together and we use this dataset further for analysis.

Semi-supervised algorithms are used in cases where dataset is partially available like images in medical industry. If we are creating a cancer detection solution by analysing the images of the patients, most probably we will not have enough sample set of training images. Here, semi-supervised approach can be helpful.

Now we will discuss the last category in machine learning called reinforcement learning.

1.4.4 Reinforcement learning

Image you are playing a game of chess with a computer. And it goes like this:

Round 1: You win after 5 moves

Round 2: You win after 8 moves

Round 3: you win after 14 moves

Round 4: you win after 21 moves

Round 5: computer wins!

What is happening here is, the algorithm is training itself iteratively depending on each interaction and correcting/improving itself.

Formally, reinforcement learning solutions are self-sustained solutions which train themselves using a sequence of trial and error. One sequence follows the other. The heart of reinforcement learning are reward signals. If the action is positive then the reward is positive indicating to continue on it. If the action is negative, the reward will penalise the activity. Hence, the solution will always correct itself and move ahead thereby improving itself iteratively.

Self-driving cars are the best examples for reinforcement learning algorithms. They detect when they have to turn left or right, when to move and when to stop. Modern video games also employ reinforcement learning algorithms. Reinforcement learning is allowing us to break the barriers of technology and imagine things which were earlier thought impossible.

With this we have covered the different types of machine learning algorithms. Together, they are harnessing the true power of data and creating a long-lasting impact on our lives.

But the heart of the solutions is the technology, which we have not discussed yet. We will now move to the technology stack required to make these solutions tick.

1.5 Technical toolkit

The following tools are used for different facets of the project:

1. Data Engineering: Hadoop, Spark, Scala, Java, C++, SQL, AWS Redshift, Azure
2. Data Analysis: SQL, R, Python, Excel
3. Machine Learning: SQL, R, Python, Excel, Weka, Julia, MATLAB, SPSS, SAS
4. Visualization: Tableau, Power BI, Qlik, COGNOS
5. Model deployment: docker, flask, Amazon S3
6. Cloud Services: Azure, AWS, GCP

In this book, we are going to work using Python. You are advised to install latest version of Python on your system. Python version of (3.5+) is advisable. We will be using Jupyter Notebook, hence it is advised to install

Anaconda on your system.

Note:

All the codes and datasets will be checked-in at the GitHub repository. You are expected to replicate them and try to reproduce the results.

A most common question is: which is better R or Python? Both are fantastic languages. Both are heavily used. But recently after the introduction of TensorFlow, keras libraries on Artificial Intelligence the balance has slightly tilted in the favour of Python.

With this, we conclude the discussion on technology. Technology along with the concepts make machine learning algorithms work for us. We will be exploring all of such finer aspects throughout the book.

Congratulations ! you have completed your very first step in your journey towards learning unsupervised machine learning techniques. It is time to wrap up and move to the summary.

1.6 Summary

Machine learning and artificial intelligence are indeed path-breaking. They are changing the way we travel, we order food, we plan, we buy, we see a doctor or order prescriptions – they are making a "dent" everywhere. Machine learning is indeed a powerful capability which is paving the path for the future and is proving much better than existing technology stacks when it comes to pattern identification, anomaly detection, customizations and automation of tasks. Autonomous driving, cancer detection, fraud identification, facial recognition, image captioning, chat-bots are only a few examples where machine learning and artificial intelligence are outperforming traditional technologies. And now is the best time to enter this field. This sector is attracting investments from almost all the business functions. The field has created tons of job opportunities across the spectrum. Incredible and impressive indeed!

At the same time, the field lacks trained professionals – data analysts, data

engineers, visualisation experts, data scientists and data practitioners. They all are a rare breed now. The field requires a regular supply of budding talents who will become the leaders of tomorrow and will take data-driven-decisions. But we only scratched the surface in understanding the power of data – there are still miles to be covered.

In this introductory chapter of this book, we introduced concepts of machine learning, data science to you. We compared various processes, what are steps in a data science project, team required for it. We examined types of machine learning algorithms with their respective use cases with an emphasis on unsupervised learning algorithms.

In the following chapter, we will dive deeper into unsupervised learning concepts of clustering. All the mathematical and statistical foundations, pragmatic case study, Python implementation are being discussed. The second chapter deals with easier clustering algorithms – kmeans clustering, hierarchical clustering and DBSCAN. In the later chapters of the book we will study more complex clustering topics like GMM clustering, time series clustering, fuzzy clustering etc.

You can now proceed to the question section now!

Questions

Q1: Why is machine learning so powerful that it is being used very heavily now?

Q2: What are the different types of machine learning algorithms and how are they different from each other?

Q3: What are the steps in a machine learning project?

Q4: What is the role of data engineering and why is it important?

Q5: What are the various tools available for machine learning?

2 Clustering techniques

"Simplicity is the ultimate sophistication – Leonardo da Vinci"

Nature loves simplicity. And teaches us to follow the same path. Most of the time, our decisions are simple choices. Simple solutions are easier to comprehend, are less time consuming, painless to maintain and ponder over. The machine learning world is no different. An elegant machine learning solution is not the one which is the most complicated algorithm available but which solves the business problem. A robust machine learning solution is easier to decipher and pragmatic enough to implement. A fully functional machine learning solution cracks the business challenge effectively and efficiently and is deployable is a production environment. As a data scientist, we always strive to attain a mature, effective and scalable machine learning solution.

Recall from Chapter 1, where we discussed data and its types, nuts and bolts of machine learning and different types of algorithms available – we started with defining unsupervised learning. We also studied the steps followed in an unsupervised learning solution. Continuing on the same path, in this second chapter we are going to start our study on unsupervised learning based clustering algorithms.

We will define clustering first and then study the different types of clustering techniques. We will examine the mathematical foundation, accuracy measurements and pros and cons of each algorithm. We will implement three of these algorithms using Python code on a dataset to complement the theoretical knowledge. The chapter ends with the various use cases of clustering techniques in the pragmatic business scenario to prepare for the actual business world. This technique is being followed throughout the book where we study the concepts first, implement the actual code to enhance the Python skills and dive into the real-world business problems.

We are going to study basic clustering algorithms in this chapter which are kmeans clustering, hierarchical clustering and DBSCAN clustering. These

clustering algorithms are generally the starting points whenever we want to study clustering. In the later chapters of the book, we are going to explore more complex algorithms like spectrum clustering, Gaussian Mixture Models, time series clustering, fuzzy clustering etc. If you have a good understanding of kmeans clustering, hierarchical clustering and DBSCAN – you can skip to the next chapter. Still, it is advisable to read the chapter once – you might find something useful to refresh your concepts!

In this second chapter, we are going to cover the following topics:

1. Clustering techniques and salient use cases in the industry
2. Various clustering algorithms available
3. K-means, hierarchical clustering and DBSCAN clustering
4. Implementation of algorithms in Python
5. Case study on cluster analysis
6. Summary

Let's first understand what we mean by "*clustering*". All the very best on your journey to master unsupervised learning based clustering techniques!

2.1 Technical toolkit

We are using the 3.6+ version of Python in this chapter. A basic understanding of Python and code execution is expected. You are advised to refresh concepts of object-oriented programming and Python concepts.

Throughout the book, we are using Jupyter notebook to execute the code. Jupyter offers flexibility in execution and debugging, hence it is being used. It is quite user-friendly and is platform or operating system agnostic. So if you are using Windows or Mac OS, Jupyter should work just fine.

All the datasets and code files are checked-in to the Github repository at (https://github.com/vverdhan/UnsupervisedLearningWithPython/tree/main/Ch You need to install the following Python libraries to execute the code – `numpy`, `pandas`, `matplotlib`, `scipy`, `sklearn`. CPU is good enough for execution, but if you face some computing lags, switch to GPU or Google Collaboratory (colab). Google colab offers free-of-cost computation for

machine learning solutions. You are advised to study more about Google Colab and how to use it for training the machine learning algorithms.

Now, we are starting with clustering in the following section.

2.2 Clustering

Consider this. A group of children is asked to group the items in a room into different segments. Each child can use their own logic. Someone might club the objects based on the weight, other children might use material while someone might use all three - weight, ,material and colour. The permutations are many and depend on the *parameters* used for grouping. Here, a child is segmenting or *clustering* the objects based on the chosen logic.

Formally put, *clustering* is used to group objects with similar attributes in the same segments, and the objects with different attributes in different segments . The resultant clusters are homogeneous within themselves while heterogeneous with each other. We can understand it better by means of a diagram below (Figure 2-1).

Figure 2-1 Clustering is grouping of objects with similar attributes into logical segments. The grouping is based on a similarity trait shared by different observations and hence they are grouped into a group
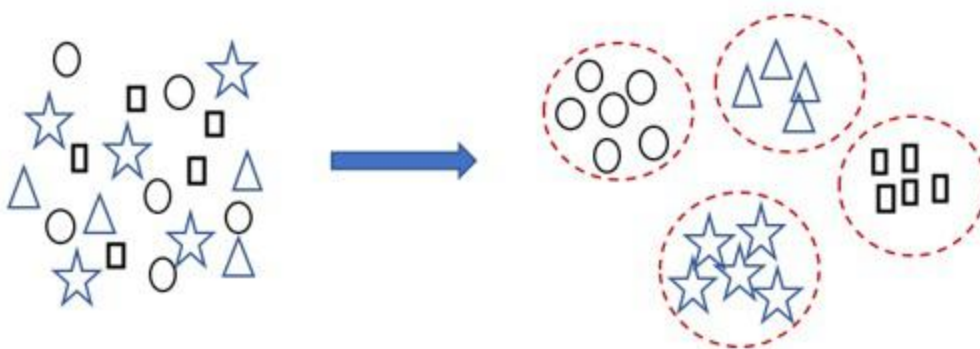
Cluster analysis is not one individual algorithm or solution, rather it is used as a problem solving mechanism in practical business scenarios. It is an iterative process following a logical approach and qualitative business inputs. It results in generating a thorough understanding of the data, logical patterns

in it, pattern-discovery and information retrieval. Being an unsupervised approach, clustering does not need a target variable . It performs segmenting by analysing underlying patterns in the dataset which are generally multi-dimensional and hence difficult to analyse with traditional methods.

Ideally we would want the clustering algorithms to have the following attributes:

1. The output clusters should be easy to explain and comprehend, usable and should make business sense. The number of clusters should not be too little or too much. For example, if we have only 2 clusters the division is not clear and decisive. Or if we have 20 clusters, the handling will become a challenge.
2. The algorithm should not be too sensitive to outliers or missing values or the noise in the dataset. Generally put, a good solution will be able to handle multiple data types.
3. A good solution will require less domain understanding for the input parameters used for the clustering purpose. It allows analysts with less domain understanding to train the clustering algorithm.
4. The algorithm should be independent of the order of the input parameters. If the order matters, the clustering is biased on the order and hence it will add more confusion to the process.
5. As we generate new datasets continuously, the clusters have to be scalable to newer training examples and should not be a time-consuming process.

As one could imagine, the clustering output will depend on the attributes used for grouping. In the (Figure 2-2) shown below, there can be two logical groupings for the same dataset and both are equally valid. Hence, it is prudent that the attributes or *variables* for clustering are chosen wisely and often it depends on the business problem at hand.

Figure 2-2 Using different attributes for clustering results in different clusters for the same dataset. Hence, choosing the correct set of attributes define the final set of results we will achieve
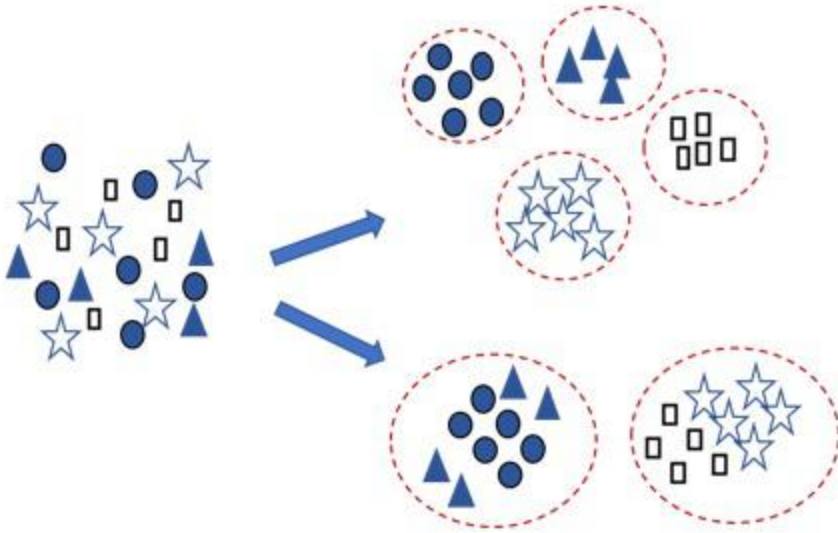
Along with the attributes used in clustering, the actual technique used also makes a lot of difference. There are quite a few (in-fact more than 80) clustering techniques researchers have worked upon. For the interested audience, we are providing a list of all the clustering algorithms in the Appendix. We are starting with understanding different clustering techniques in the next section.
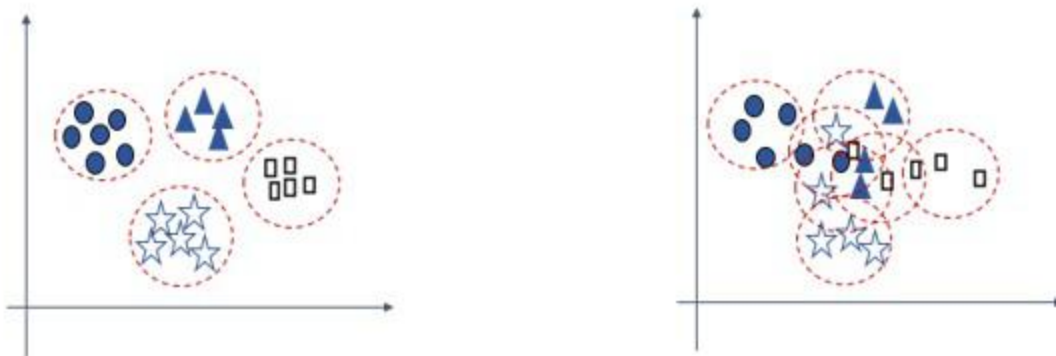
2.2.1 Clustering techniques

Clustering can be achieved using a variety of algorithms. These algorithms use different methodologies to define similarity between objects. For example, density based clustering, centroid based clustering, distribution based methods etc. Even to measure the distance between objects, there are multiple techniques like Euclidean distance, Manhattan distance etc. The choice of distance measurement leads to different similarity scores. We are going to study these similarity measurement parameters in a later section.

At a high-level we can identify two broad clustering methods: *hard clustering* and *soft clustering*. When the decision is quite clear that an object belongs to a certain class or cluster it is referred as Hard clustering. In hard clustering we are quite sure of an object's class. On the other hand, soft clustering assigns a likelihood score for an object to belong to a particular cluster. So a soft clustering method will not put an object into a cluster, rather an object can belong to multiple clusters . Soft clustering sometimes is also called *fuzzy*

clustering.

Figure 2-3 Hard clustering has distinct clusters whereas in the case of soft clustering, a data point can belong to multiple clusters and we get likelihood score for a data point to belong to a cluster

We can broadly classify the clustering techniques as shown in the (Table 2-1) below:

Table 2-1 Classification of clustering methodologies, brief descriptions and examples

S. No.	Clustering methodology	A brief description of the method	Example
1	Centroid based clustering	Distance from a defined centroid	k-means
2	Density based models	Data points are connected in dense regions in a vector space	DBSCAN, OPTICS
3	Connectivity based clustering	Distance connectivity is the modus operandi	Hierarchical clustering, BIRCH

4	Distribution models	Modelling is based on statistical distributions	Gaussian Mixture models
5	Deep learning models	Unsupervised neural network based	Self-organizing maps

The methods described in (Table 2-1) are not the only ones which are available to be used. We can have graph-based models, overlapping clustering, subspace models etc.

Generally, the popular six algorithms used in clustering in the industry are as follows:

1. K-means clustering (with variants like k-medians, k-medoids)
2. Agglomerative clustering or hierarchical clustering
3. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
4. Spectral Clustering
5. Gaussian mixture models or GMM
6. BIRCH (Balanced Iterative Reducing & Clustering using Hierarchies)

There are multiple other algorithms available like Chinese whisper, canopy clustering, SUBCLU, FLAME etc. We are studying the first three algorithms in this chapter and some of the advanced ones in subsequent chapters in the book.

🎯 POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

1. DBSCAN clustering is centroid-based clustering technique. TRUE or FALSE.

2. Clustering is a supervised learning technique having a fixed target variable. TRUE or FALSE.

3. What is the difference between hard clustering and soft clustering?

In the next section, we are starting with the centroid based clustering methods where we will study k-means clustering.

2.3 Centroid based clustering

Centroid based algorithms measure similarity of the objects based on their distance to the centroid of the clusters. The smaller the distance is, higher is the similarity. We can understand the concept by (Figure 2-3) below. The figure on the right side represents the respective centroids for each of the group of clusters.

To get more clarity on the concept of centroid and other mathematical concepts, refer to the appendix at the end.

Figure 2-3 Centroid based clustering methods create a centroid for the respective clusters and the similarity is measured based on the distance from the centroid. In this case, we have 5 centroids. And hence, we have five distinct clusters here
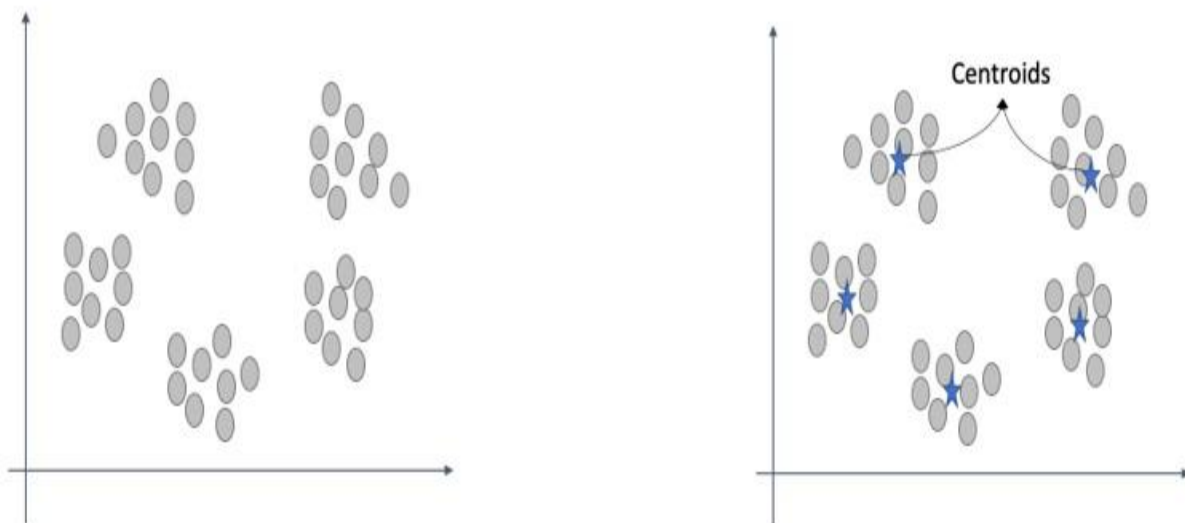
In clustering, distance plays a central part as many algorithms use it as a metric to measure the similarity. In centroid-based clustering, distance is measured between points and between centroids. There are multiple ways to measure the distance. The most widely used are listed below:

1. **Euclidean distance**: It is the most common distance metric used. It represents the straight line distance between the two points in space and is the shortest path between the two points. If we want to calculate the distance between points P_1 and P_2 where coordinates of P_1 are (x_1, y_1) and P_2 are (x_2, y_2), then Euclidean distance is given by (Equation 2-1) below. The geometric representation is shown in (Figure 2-4)

$$\text{Distance} = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \qquad \text{(Equation 2-1)}$$

2. **Chebyshev distance**: Named after Russian mathematician *Pafnuty Chebyshev*, it is defined as the distance between two points such that their differences are maximum value along any co-ordinate dimension. Mathematically, we can represent Chebyshev distance in (Equation 2-2) below and shown in (Figure 2-4):

$$\text{Distance}_{\text{Chebyshev}} = \max(|y_2 - y_1|, |x_2 - x_1|)$$
(Equation 2-2)

3. **Manhattan distance**: Manhattan distance is a very easy concept. It simply calculates the distance between two points in a grid-like path and the distance is hence measured along the axes at right angles. Hence, sometimes it is also referred to as city block distance or taxi cab metric. Mathematically, we can represent Manhattan distance in (Equation 2-3) and is shown in (Figure 2-4):

$$\text{Distance}_{\text{Manhattan}} = (|y_2 - y_1| + |x_2 - x_1|)$$
(Equation 2-3)

Manhattan distance in L1 norm form while Euclidean distance is L2 norm form. You can refer to the Appendix to study the L1 norm and L2 norm in detail. If we have high number of dimensions or variables in the dataset, Manhattan distance is a better choice than Euclidean distance. This is due to *Curse of Dimensionality* which we are studying in Chapter 3 of the book.

4. **Cosine distance**: Cosine distance is used to measure the similarity between two points in a vector-space diagram. In trigonometry, cosine

of 0 is 1 and cosine of 90^0 is 0. Hence, if two points are similar to each other, the angle between them will be zero hence cosine will be 1 which means the two points are very similar to each other, and vice versa. Mathematically, cosine similarity can be shown as (Equation 2-4). If we want to measure the cosine between two vectors A and B, then cosine is

Distance $_{cosine}$ = (A . B) / (||A|| ||B||)
(Equation 2-4)

Figure 2-3 Euclidean distance, Manhattan distance, Chebyshev distance and cosine similarity are the primary distance metrics used. Note, how the distance is different for two points using these metrics
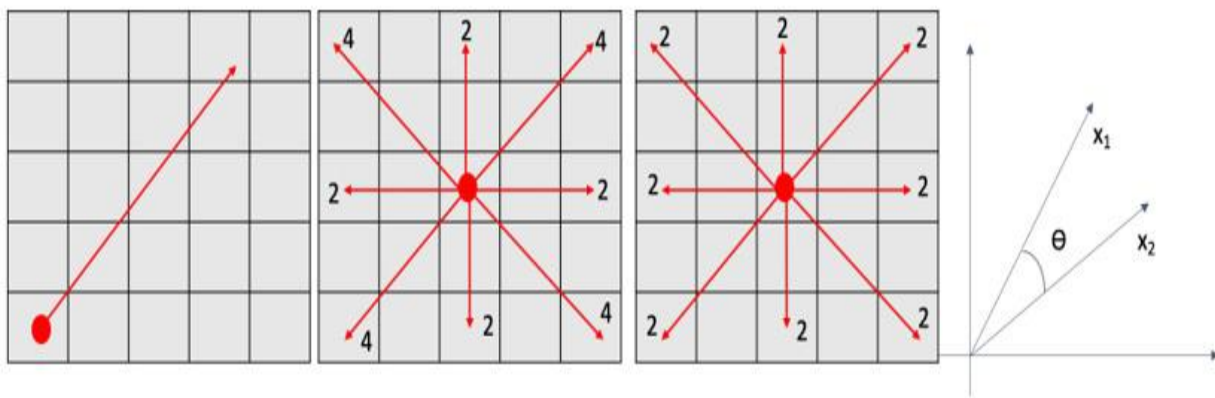
There are other distance measuring metrics like Hamming distance, Jaccard distance etc. Mostly, we use Euclidean distance in our pragmatic business problems but other distance metrics are also used sometimes.

Note:

The above distance metrics are true for other clustering algorithms too. You are advised to test the Python codes in the book with different distance metrics and compare the performance.

Now we have understood the various distance metrics, we will proceed to study k-means clustering which is the most widely used algorithm.

2.3.1 K-means clustering

k-means clustering is an easy and straightforward approach. It is arguably the most widely used clustering method to segment the data points and create non-overlapping clusters. We have to specify the number of clusters "k" we wish to create as an input and the algorithm will associate each observation to exactly one of the k clusters.

Note:

k-means clustering is sometimes confused with k-nearest neighbour classifier (knn). Though there is some relationship between the two, knn is used for classification and regression problems.

It is quite an elegant approach and starts with some initial cluster centres and then iterates to assign each observation to the closest centre. In the process the centres are re-calculated as the mean of points in the cluster. Let's study the approach used in step-by-step fashion by using the diagram in (Figure 2-4) below. For the sake of simplicity, we are assuming that there are three clusters in the dataset below.

Step 1: Let us assume that we have all the data points as shown below in Step 1.

Figure 2-4 Step 1 represents the raw data set. In step 2, the algorithm initiates random three centroids as we have given the input of a number of clusters as three. In step 3, all the neighboring points of the centroids, are assigned the same cluster
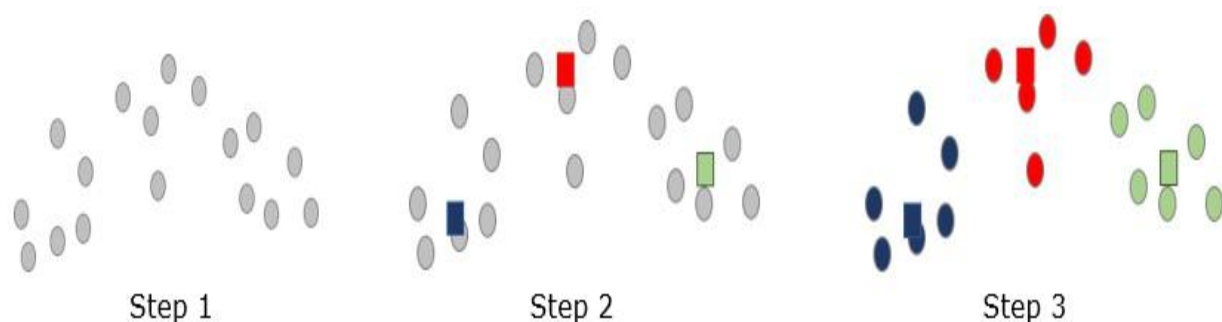

Step 2: The *three* centres are initialized randomly as shown by three squares – blue, red and green. This input of three is the final number of clusters we wish to have.

Step 3: The distance of all the data points is calculated to the centres and the points are assigned to the nearest centre. Note that the points have attained blue, red and green colours as they are nearest to those respective centres.

Step 4: The three centres are re-adjusted in this step. The centres are re-calculated as the mean of the points in that cluster as shown in (Figure 2-5) below. We can see that in Step 4, the three squares have changed their respective positions as compared to Step 3.

Figure 2-5 The centroids are re-calculated in step 4. In step 5, the data points are again re-assigned new centers. In step 6, the centroids are again re-adjusted as per the new calculations
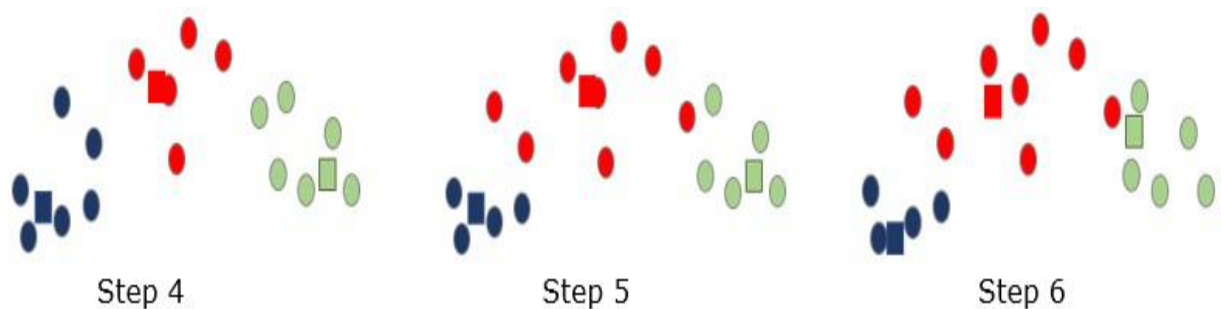
Step 5: The distance of all the data points is re-calculated to the new centres and the points are reassigned to the nearest centres again. Note that two blue data points have become red while a red point has become green in this step.

Step 6: The centres are again re-adjusted as they were done in Step 4.

Figure 2-6 The centroids are re-calculated and this process continues till we can no longer improve the clustering. And then the process stops as shown in step 8
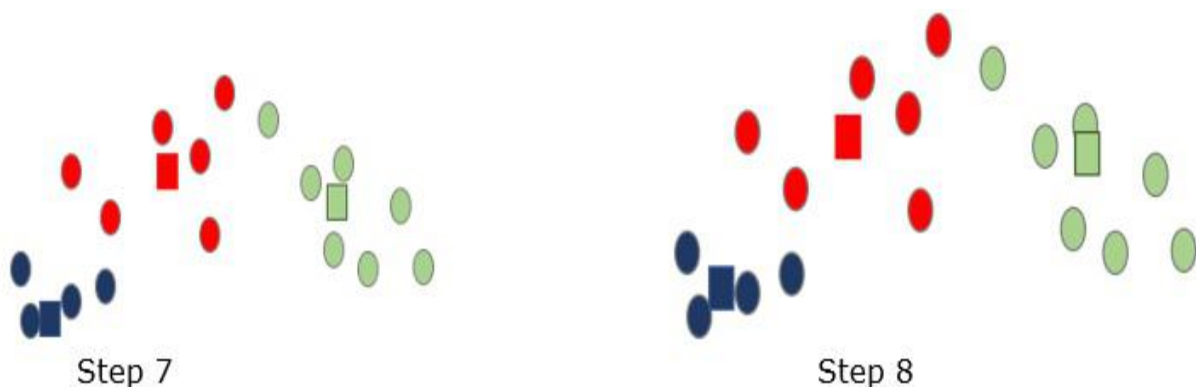
Step 7: The data points are again assigned a new cluster as shown in (Figure 2-6) above.

Step 8: And the process will continue till the convergence is achieved. In other words, the process continues till there is no more reassignment of the data points. And hence, we cannot improve the clustering further and the final clustering is achieved.

The objective of k-means clustering is to ensure that the within-cluster variation is as small as possible while the difference between clusters is as big as possible. In other words, the members of the same cluster are most similar to each other while members in different clusters are dissimilar. Once the results no longer change, we can conclude that a local optimum has been reached and clustering can stop. Hence, the final clusters are homogeneous within themselves while heterogeneous with each other.

It is imperative to note two points here:

1. Since k-means clustering initializes the centres randomly, hence it finds a local optimum solution rather than a global optimum solution. Hence, it is advisable to iterate the solution multiple times and choose the best output from all the results.
2. We have to input the number of final clusters "k" we wish to have and it changes the output drastically. A very small value of k will result in redundant clusters as there will not be any use. Having a very high value of k, will create clusters which are different from each other minutely. Moreover, having a very high number of clusters will be difficult to manage and refresh in the long run. Let's study by an example. If a telecom operator has 1 million subscribers, then if we take number of clusters as 2 or 3, the resultant cluster size will be very large. It can also lead to similar customers classified in the same segment. On the other hand, if we take the number of clusters as 50 or 60, due to the sheer number of clusters – the solution becomes unmanageable to manage and maintain.

With different values of "k" we get different results, hence it is necessary that we understand how we can choose the optimum number of clusters for a dataset. But before that, we have to examine the parameters to assess the

accuracy of the clustering algorithms because it is an input to find the optimum number of clusters.

2.3.2 Measure the accuracy of clustering

The objective of clustering is to find the most clean clusters. Ideally, if we have the same number of clusters as the number of observations the results will be completely accurate. In other words, if we have 1 million customers, the purest clustering will have 1 million clusters – wherein each customer is in a separate cluster. But it is not the best approach and is not a pragmatic solution. Clustering intends to create group of similar observations in one cluster and we use the same principle to measure the accuracy of our solution.
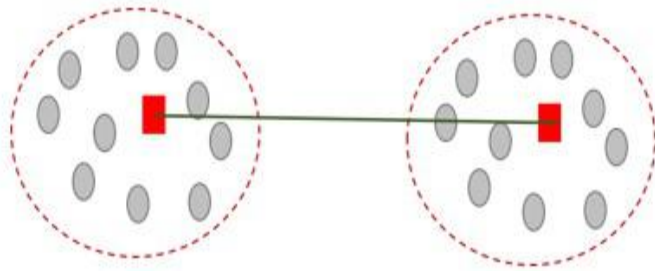
1. **With the cluster sum of squares (WCSS) or Cohesion**: This index measures the variability of the data points with respect to the distance they are from the centroid of the cluster. Here, first we calculate the distance between a data point and the centroid to the respective cluster. This process is done for all the data points in that cluster. The results are added and then divided by the total number of points in that cluster. The process is repeated for all the clusters. And finally, the results are averaged across all the clusters. If the value is too large, it shows there is a large data spread whereas the smaller value indicates that the data points are quite similar and homogeneous and hence the cluster is compact.

Sometimes, this intracluster distance is also referred to as *inertia* for that cluster. It is simply the summation of all the distances. Lower the value of inertia, better the cluster is.

Figure 2-7 Intra cluster vs inter cluster distance – both are used to measure the purity of the final clusters and the performance of the clustering solution
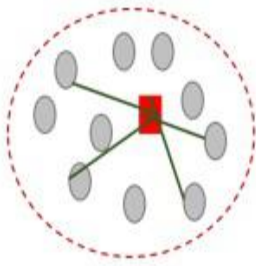
Within Cluster or Intra cluster   Inter cluster distance between two clusters

2. **Inter cluster sum of squares**: This metric is used to measure the distance between centroids of all the clusters. To get it, we measure the distance between centroids of all the clusters and divide it by the number of clusters to get the average value. The bigger it is, better is the clustering indicating that clusters are heterogeneous and distinguishable from each other. It is shown in (Figure 2-7) above.

3. **Silhouette Value** is one of the metrics used to measure the success of clustering. It ranges from -1 to +1 and a higher value is better. It measures how an observation is similar to the objects in its own cluster as compared to other clusters. As a first step, for each observation - we calculate the average distance from all the data points in the same cluster, let's call is x_i. Then we calculate the average distance from all the data points in the nearest cluster, let's call it y_i. We will then calculate the coefficient by the equation (Equation 2-5) below

Silhouette Coefficient = $(y_i - x_i)/ \max (y_i, x_i)$
(Equation 2-5)

If the value of coefficient is -1, it means that the observation is in the wrong cluster.

If it is 0, the observation is very close to the neighboring clusters.

If the values of coefficient +1, it means that the observation is at a distance from the neighboring clusters.

Hence, we would expect to get the highest value for the coefficient to have a good clustering solution.

4. **Dunn Index** can also be used to measure the efficacy of the clustering. It uses the KPI defined in point 2 and point 3 above and is given by the (Equation 2-6) below

Dunn Index = min (Inter cluster distance)/max (Intra cluster distance)   (Equation 2-6)

Clearly, we would strive to maximize the value of Dunn index. To achieve it, the numerator should be as big as possible implying that clusters are at a distance from each other, while the denominator should be as low as possible signifying that the clusters are quite robust and close-packed.

Now we have examined the methods to measure the performance of our algorithm. We will now move to find out the best value of "k" for k-means clustering.

2.3.3 Finding the optimum value of "k"

Choosing the most optimum number of clusters is not easy. As we have said earlier, the finest clustering is when the number of clusters equals the number of observations – but as we studied in the last section, it is not practically possible. But we have to provide the number of clusters "k" as an input to the algorithm.

Figure 2-8 Elbow method to find the optimal number of clusters. The red circle shows the kink. But the final number of clusters is dependent on business logic and often we merge/split clusters as per business knowledge. Ease to maintain the clusters also plays a crucial role in the same

Selecting k with the Elbow Method
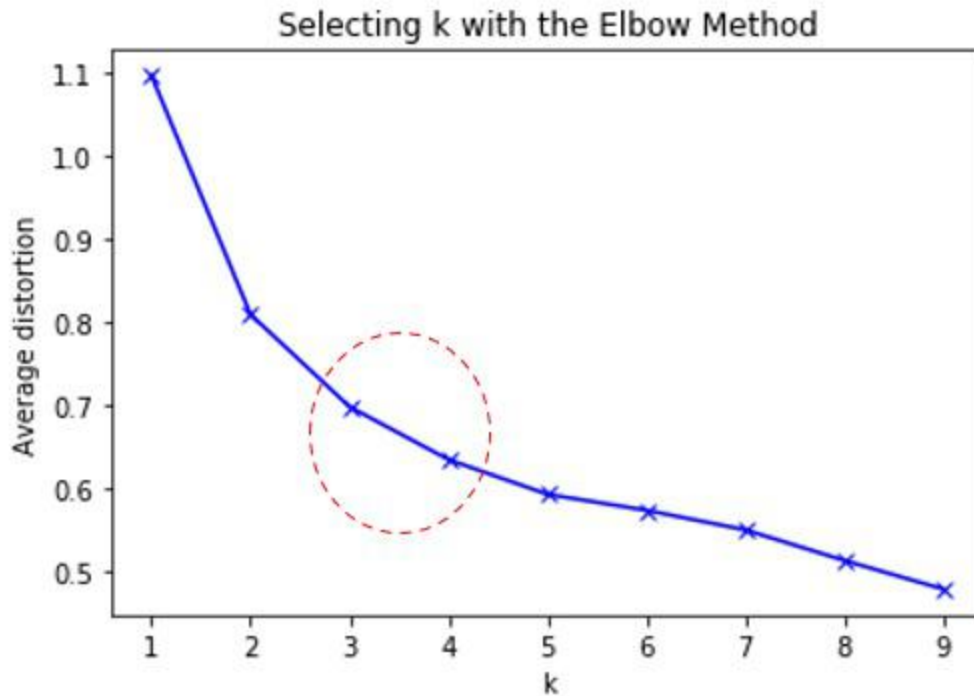
Perhaps the most widely used method for finding the optimum value of "k" is the *Elbow Method.* In this method, we calculate within the cluster sum of squares or WCSS for different values of "k". The process is the same as discussed in the last section. Then, WCSS is plotted on a graph against different values of "k". Wherever we observe a kink or elbow, as shown in (Figure 2-8), it is the most optimum number of clusters for the dataset. Notice the sharp edge depicted in (Figure 2-8).

◎ POP QUIZ – answer these question to check your understanding..
Answers at the end of the book

1. K-means clustering does not require number of clusters as an input-TRUE or FALSE

2. Knn and k-means clustering are one and the same thing – TRUE or FALSE

3. Describe the process to find the most optimal value of "k"

But it does not mean that it is the final number of clusters we suggest for the business problem. Based on the number of observations falling in each of the clusters, a few clusters might be combined or broken into sub-clusters. We also consider the computation cost required to create the clusters. Higher the number of clusters, greater is the computation cost and the time required.

We can also find the optimum number of clusters using the Silhouette Coefficient we discussed earlier.

Note:

It is imperative that business logic of merging a few clusters or breaking a few clusters is explored. Ultimately, the solution has to be implemented in real-world business scenarios.

With this, we have examined nuts and bolts of k-means clustering – the mathematical concepts and the process, the various distance metrics and determining the best value of k. We will now study the advantages k-means algorithm offers to us.

2.3.4 Pros and cons of k-means clustering

k-means algorithm is quite a popular and widely implemented clustering solution. The solution offers the following advantages:

1. It is simple to comprehend and relatively easier to implement as compared to other algorithms. The distance measurement calculation makes it quite intuitive to understand even by users from non-statistics backgrounds.
2. If the number of dimensions is large, k-means algorithm is faster than other clustering algorithms and creates tighter clusters. It is hence preferred if the number of dimensions are quite big.
3. It quickly adapts to new observations and can generalize very well to clusters of various shapes and sizes.
4. The solution produces results through a series of iterations of re-calculations. Most of the time the Euclidean distance metric is used which makes it less computationally expensive. It also ensures that the

algorithm surely converges and produces results.

K-means is widely used for real life business problems. Though there are clear advantages of k-means clustering, we do face certain challenges with the algorithm:

1. Choosing the most optimum number of clusters is not easy. We have to provide it as an input. With different values of "k", the results will be completely different. The process to choose the best value of "k" is explored in the next section.
2. The solution is dependent on the initial values of centroids. Since the centroids are initialized randomly, the output will be different with each iteration. Hence, it is advisable to run multiple versions of the solution and choose the best one.
3. The algorithm is quite sensitive to outliers. They can mess up the final results and hence it is imperative that we treat outliers before starting with clustering. We can also implement other variants of k-means algorithm like *k-modes* clustering to deal with the issue of outliers. We are discussing dealing with outliers in subsequent chapters.
4. Since the basic principle of k-means clustering is to calculate the distance, hence the solution is not directly applicable for categorical variables. Or in other words, we cannot use categorical variables directly, since we can calculate the distance between numeric values but cannot perform mathematical calculations on categorical variables. To resolve it, we can convert categorical variables to numeric ones using one-hot encoding which we are discussing towards the end of this chapter.

Despite these problems, k-means clustering is one of the most used clustering solutions owing to its simplicity and ease to implement. There are different implementations of k-means algorithm like k-medoids, k-median etc. which are sometimes used to resolve the problems faced.

5. As the name suggests, **k-median clustering** is based on medians of the dataset as compared to centroid in k-means. This increases the amount of computation time as median can be found only after the data has been sorted. But at the same time, k-means is sensitive to outliers whereas k-medians is less affected by them.

6. Next, we have **k-medoids clustering** as one of the variants of the k-means algorithm. Medoids are similar to means except they are always from the same dataset and are implemented when it is difficult to get means like images. A medoid can be thought as the most central point in a cluster which is least dissimilar to all the other members in the cluster. K-medoids choose the actual observations as the centers as compared to k-means where the centroids may not even be part of the data. It is less sensitive to outliers as compared to k-means clustering algorithm.

There are other versions too like kmeans++, mini-batch k-means etc. Moreover, having different distance measurement metrics may produce different results for k-means algorithm.

This section concludes our discussion on k-means clustering algorithm. It is time to hit the lab and develop actual Python code!

2.3.5 k-means clustering implementation using Python

We will now create a Python solution for k-means clustering. In this case, we are using the dataset from (link). The details of the data can be found here.

This dataset has information about features of four models of cars. Based on the features of the car, we are going to group them into different clusters.

Step 1: Import the libraries and the dataset into a dataframe. Here, vehicles.csv is the input data file. If the data file is not in the same folder as the Jupyter notebook, you would have to provide the complete path to the file. Dropna is used to remove the missing values, if any.

```
import pandas as pd
vehicle_df = pd.read_csv('vehicle.csv').dropna()
```

Step 2: Perform some initial checks on the data, like shape, info, top five rows, distribution of classes etc. This is to ensure that we have loaded the complete dataset and there is no corruption while loading the dataset. `Shape` command will give the number of rows and columns in the data, `info` with describe all the variables and their respective types and `head` will display the first 5 rows. The `value_counts` displays the distribution for the `class`

variable. Or in other words, `value_counts` returns the count of the unique values.

```
vehicle_df.shape
vehicle_df.info()
vehicle_df.head()
pd.value_counts(vehicle_df['class'])
```

Step 3: Let's generate two plots for the variable "`class`". The dataset has more examples from car while for bus and van it is a balanced data. We have used matplotlib library to plot these graphs. The output of the plots are shown below.

```
import matplotlib.pyplot as plt
%matplotlib inline
pd.value_counts(vehicle_df["class"]).plot(kind="bar")
pd.value_counts(vehicle_df['class']).hist(bins=300)
```
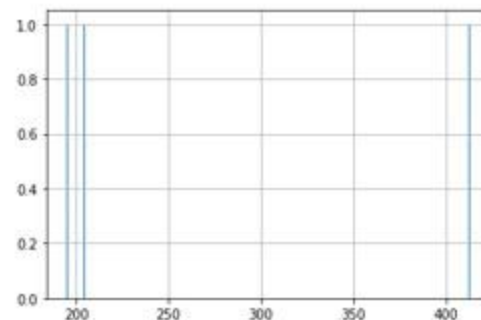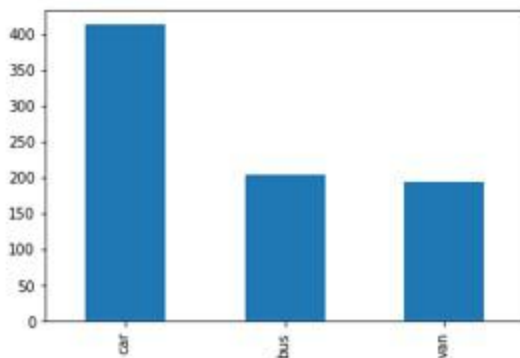

Step 4: We will now check if there are any missing data points in our dataset. There are no missing data points in our dataset as we have already dealt with them.

```
vehicle_df.isna().sum()
```

Note:

We will be discussing the methods to deal with missing values in later chapters as dropping the missing values is generally not the best approach.

Step 5: We will standardise our dataset now. It is a good practice to standardise the dataset for clustering. It is important as the different

dimensions might be on a different scale. This is done using `zscore` and
`StandardScaler()` function below. Refer to the appendix of the book to
examine the difference between `zscore` and `StandardScaler()` function.

```
vehicle_df_1 = vehicle_df.drop('class', axis=1)
from scipy.stats import zscore
vehicle_df_1_z = vehicle_df_1.apply(zscore)
from sklearn.preprocessing import StandardScaler
import numpy as np
sc = StandardScaler()
X_standard = sc.fit_transform(vehicle_df_1)
```

Step 6: We will now have a quick look at the dataset by generating a scatter
plot. The plot displays the distribution of all the data points we have created
as `X_standard` in the last step.

```
plt.scatter(X_standard[:,0], X_standard[:,1])
plt.show()
```
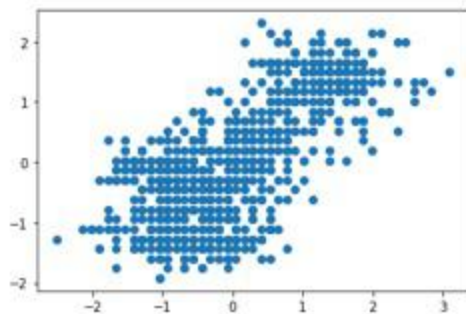

Step 7: We will now perform k-means clustering. First, we have to select the
optimum number of clusters using the elbow-method. From sklearn library,
we are importing KMeans. In a for loop, we iterate for the values of clusters
from 1 to 10. In other words, the algorithm will create 1, 2,3, 4 upto 10
clusters and will then generate the results for us to choose the most optimal
value of k.

In the code snippet below, model object contains the output of KMeans
algorithm which is then fit on the X_standard generated in the last step. Here,
Euclidean distance has been used as a distance metric.

```
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
```

```
clusters=range(1,10)
meanDistortions=[]
for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(X_standard)
    prediction=model.predict(X_standard)
    meanDistortions.append(sum(np.min(cdist(X_standard, model.clu
                         .shape[0])
plt.plot(clusters, meanDistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```
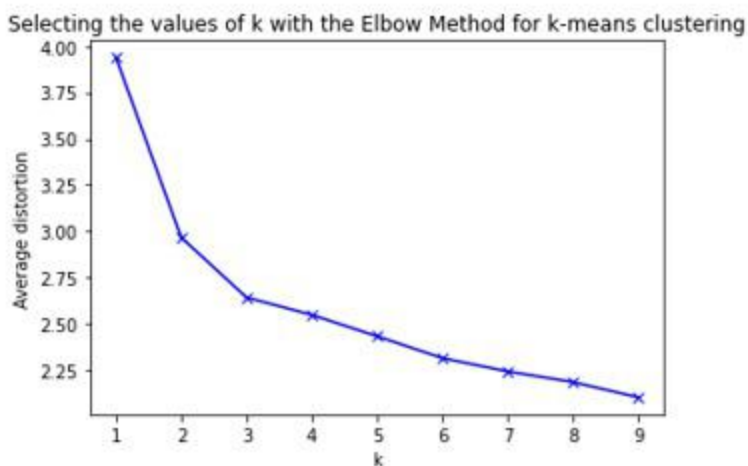

Step 8: As we can observe, optimal number of clusters as 3. It is the point, where we can observe a sharp kink in the graph. We will continue with k-means clustering with a number of clusters as 3. `random_state` is a parameter which is used to determine random numbers for centroid initialization. We are setting it to a value to make randomness deterministic.

```
kmeans = KMeans(n_clusters=3, n_init = 15, random_state=2345)
kmeans.fit(X_standard)
```

Step 9: Get the centroids for the clusters

```
centroids = kmeans.cluster_centers_
centroids
```

Step 10: Now we are using the centroids so that they can be profiled by the columns.

```
centroid_df = pd.DataFrame(centroids, columns = list(X_standard)
```

Step 11: We will now create a `dataframe` only for the purpose of creating the labels and then we are converting it into categorical variables.

```
dataframe_labels = pd.DataFrame(kmeans.labels_ , columns = list([
dataframe_labels['labels'] = dataframe_labels['labels'].astype('c
```

Step 12: In this step, we are joining the two `dataframes`

```
dataframe_labeled = vehicle_df_1.join(dataframe_labels)
```

Step 13: A group by is done to create a data frame requires for the analysis

```
dataframe_analysis = (dataframe_labeled.groupby(['labels'] , axis
dataframe_labeled['labels'].value_counts()
```

Step 15: Now, will create a visualization for the clusters we have defined. This is done using the `mpl_toolkits` library. The logic is simple to understand. The data points are coloured as per the respective labels. The rest of the steps are related to the display of plot by adjusting the label, title, ticks etc. Since it is not possible to plot all the 18 variables in the plot, we have chosen 3 variables to show in the plot.

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(8, 6))
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=20, azim=60)
kmeans.fit(vehicle_df_1_z)
labels = kmeans.labels_
ax.scatter(vehicle_df_1_z.iloc[:, 0], vehicle_df_1_z.iloc[:, 1],
ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('Length')
ax.set_ylabel('Height')
ax.set_zlabel('Weight')
ax.set_title('3D plot of KMeans Clustering on vehicles dataset')
```

3D plot of KMeans Clustering on vehicles dataset
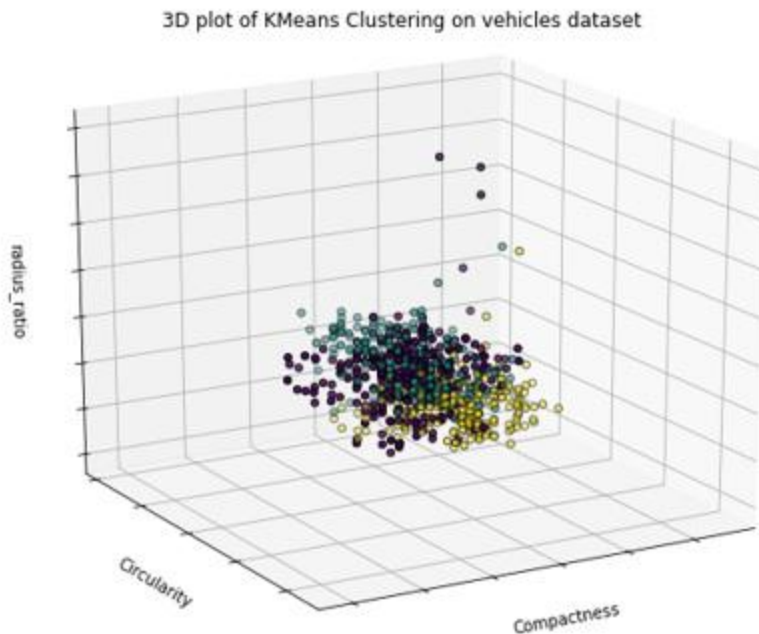
We can also test the above code with multiple other values of k. We have created the code with different values of k. In the interest of space, we have put the code of testing with different values of k at the github location.

In the example above, we first did a small exploratory analysis of the dataset.

Note:

Exploratory data analysis (EDA) holds the key to a robust machine learning solution and a successful project. In the subsequent chapters, we will create detailed EDA for datasets.

It was followed with identifying the optimum number of clusters which in this case comes out to be three. Then we implemented k-means clustering. You are expected to iterate the k-means solution with different initializations and compare the results, iterate with different values of k and visualize to analyze the movements of data points. We will be using the same dataset later in the chapter where we will create hierarchical clustering using Python.

Centroid based clustering is one of the most recommended solutions owing to its less complicated logic, ease to implement, flexibility and trouble-free maintenance. Whenever we require clustering as a solution, mostly we start

with creating a k-means clustering solution which acts as a benchmark. The algorithm is highly popular and generally one of the first solutions utilized for clustering. Then we test and iterate with other algorithms.

This marks the end of discussion on centroid-based clustering algorithms. We will now move forward to connectivity-based solutions and discuss hierarchical clustering in the next section.

2.4 Connectivity based clustering

"Birds of the same feather flock together" is the principle followed in connectivity based clusters. The core concept is - objects which are closer to each other are similar to each other. Hence, based on the distance between these objects they are clubbed into clusters. An example of such a representation is shown in (Figure 2-9) below where we can iteratively group observations. As an example, we are initiating with all things, dividing into living and non-living and so on. Such representation is better shown using the diagram on the right, called the *Dendrogram.*

Figure 2-9 Hierarchical clustering utilizes grouping similar objects iteratively. On the right, we have the visual representation of the clustering called dendrogram
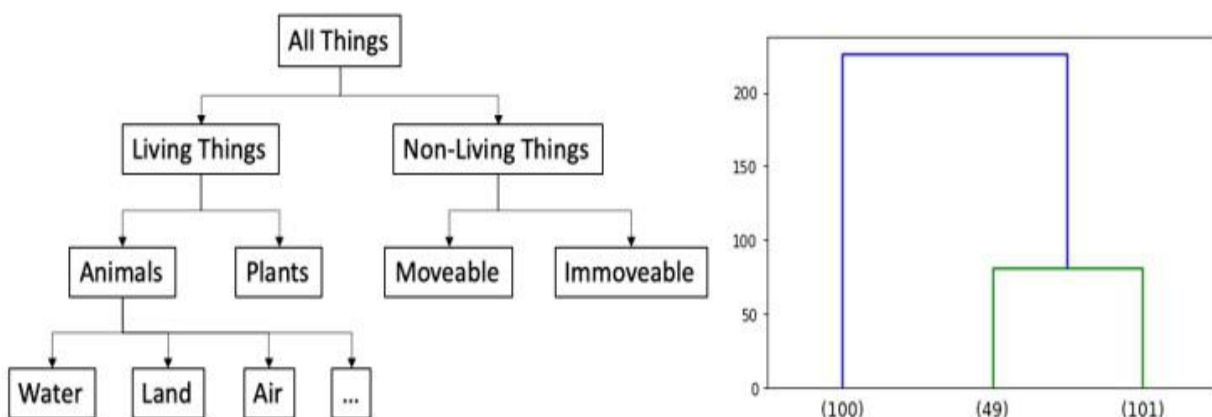

Since there is a tree-like structure, connectivity based clustering is sometimes referred as *Hierarchical* clustering.

Hierarchical clustering fits nicely into human intuition, and hence is easy to comprehend by us. Unlike k-means clustering, in hierarchical clustering we

do not have to input the number of final clusters but the method does require a termination condition i.e. when the clustering should stop. At the same time, hierarchical clustering does not suggest the optimum number of clusters. From the hierarchy/ dendrogram generated we have to choose the best number of clusters ourselves. We will explore more on it when we create the Python code for it in subsequent sections.

Hierarchical clustering can be understood by means of (Figure 2-10) below. Here the first node is the root, which is then iteratively split into nodes and subnodes. Whenever a node cannot be split further, it is called a terminal node or *leaf*.

Figure 2-10 Hierarchical clustering has a root which splits into nodes and subnodes. A node which cannot be split further is called the leaf. In bottom-up approach, merging of the leaves will take place
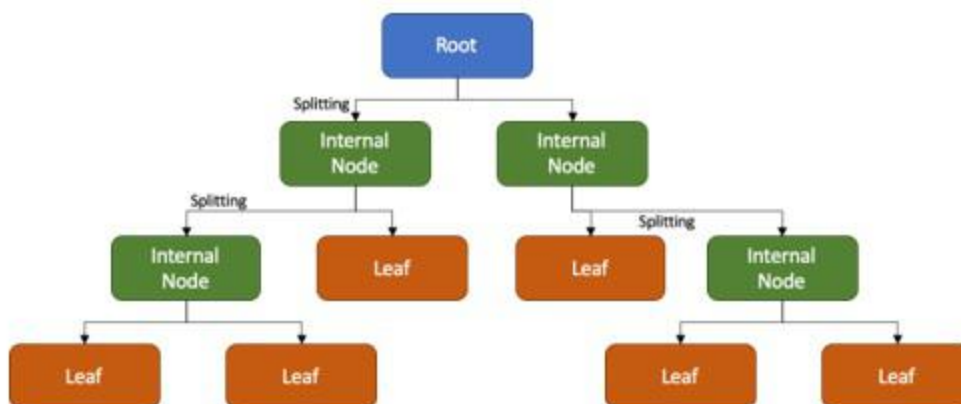
Since there are more than one process or logic to merge the observations into clusters, we can generate a large number of dendrograms which is given by the (Equation 2-7) below:

Number of dendrograms = $(2n-3)!/[2^{(n-2)} (n-2)!]$ (Equation 2-7)

where n is the number of observations or the leaves. So if we have only 2 observations, we can have only 1 dendrogram. If we have 5 observations, we can have 105 dendrograms.

Hierarchical clustering can be further classified based on the process used to

create grouping of observations, which we are exploring next.

2.4.1 Types of hierarchical clustering

Based on the strategy to group, hierarchical clustering can be subdivided into two types: *agglomerative* clustering and *divisive* clustering.

S.No.	Agglomerative methodology	Divisive methodology
1	Bottom-up approach	Top-down approach
2	Each observation creates its own cluster and then merging takes place as the algorithm goes up	We start with one cluster and then observations are iteratively split to create a tree-like structure
3	Greedy approach is followed to merge	Greedy approach is followed to split
4	An observation will find the best pair to merge and the process completes when all the observations have merged with each other	All the observations are taken at the start and then based on division conditions, splitting takes place until all the observations are exhausted or the termination condition is met

Figure 2-11 Step followed in hierarchical clustering. Left-to-right we have agglomerative clustering (splitting of the nodes) while right-to-left we have divisive clustering (merging of the nodes)

Step 4 ── Step 3 ── Step 2 ── Step 1 ──

Divisive clustering

Step 1 ── Step 2 ── Step 3 ── Step 4

Agglomerative clustering
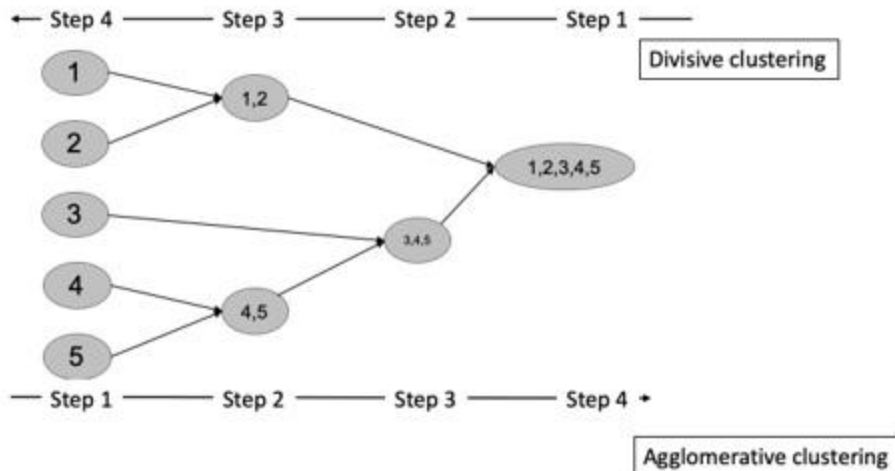
Let's explore the meaning of *greedy approach* first. Greedy approach or greedy algorithm is any algorithm which makes a best choice at each step without considering the impact on the future states. In other words, we live in-the-moment and choose the best option from the available choices at that moment. The current choice is independent of the future choices and the algorithm will solve the subproblems later. Greedy approach may *not* provide the most optimal solution but still provides a locally optimal solution which is closer to the most optimal solution in a reasonable time. Hierarchical clustering follows this greedy approach while merging or splitting at a node.

We will now examine the steps followed in hierarchical clustering approach:

Step 1: As shown in (Figure 2-11) above, let us say we have five observations in our data set – 1, 2, 3, 4 and 5.

Step 2: In this step, observation 1 and 2 are grouped into one, 4 and 5 are clubbed in one. 3 is not clubbed in any one.

Step 3: Now in this step, we group the output of 4,5 in the last step and observation 3 into one cluster.

Step 4: The output from step 3 is clubbed with the output of 1,2 as a single cluster.

In this approach, from left-to-right, we have an agglomerative approach and from right-to-left a divisive approach is represented. In an agglomerative

approach, we are merging the observations while in a divisive approach we are splitting the observations. We can use both agglomerative or divisive approaches for hierarchical clustering. Divisive clustering is an exhaustive approach and sometimes might take more time than the other.

Similar to k-means clustering, the distance metric used to measure plays a significant role here. We are aware and understand how to measure the distance between data points but there are multiple methods to define that distance which we are studying now.

2.4.2 Linkage criterion for distance measurement

We are aware that we can use Euclidean distance or Manhattan distance or Chebyshev distance etc. to measure the distance between two observations. At the same time, we can employ various methods to define that distance. And based on this input criterion, the resultant clusters will be different. The various methods to define the distance metric are:

1. **Nearest neighbours or single linkages** use the distance between the two nearest points in different clusters. The distance between the closest neighbours in distinct clusters is calculated and it is used to determine the next split/merging.
2. **Farthest neighbour or complete linkage** is opposite of the nearest neighbour approach. Here, instead of taking the nearest neighbours we concentrate on most-distant neighbours in different clusters. In other words, we determine the distance between the clusters is calculated by the greatest distance between two objects.
3. **Group average linkage** calculates the average of distances between all the possible pairs of objects in two different clusters.
4. **Ward linkage** method aims to minimize the variance of the clusters which are getting merged into one.

We can use these options of distance metrics while we are developing the actual code for hierarchical clustering, and compare the accuracies to determine the best distance metrics for the dataset. During the algorithm training, the algorithm merges the observations which will minimize the linkage criteria chosen.

Note:

Such inputs to the algorithm are referred to as hyper-parameters. These are the parameters we feed to the algorithm to generate the results as per our requirement.

We can visualise the various linkages in (Figure 2-12) below.

Figure 2-12 (i) Single linkage is for closest neighbors (ii) Complete linkage is for farthest neighbors and (iii) Group average is for average of the distance between clusters
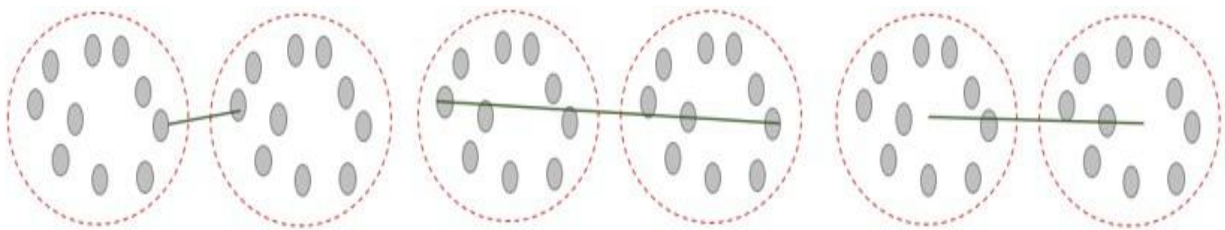
With this, we have understood the working mechanisms in hierarchical clustering. But we have still not addressed the mechanism to determine the optimum number of clusters using hierarchical clustering, which we are examining next.

2.4.3 Optimal number of clusters

Recall in k-means clustering we have to give the number of clusters as an input to the algorithm. We use elbow method to determine the optimum number of clusters. In the case of hierarchical clustering, we do not have to specify the number of clusters to the algorithm, but still we have to identify the number of final clusters we wish to have. We use a dendrogram to answer that problem.

Let us assume that we have 10 data points in total at the bottom of the chart as shown in (Figure 2-13). The clusters are merged iteratively till we get the one final cluster at the top. The height of the dendrogram at which two clusters get merged with each other represents the respective distance between the said clusters in the vector-space diagram.

Figure 2-13 Dendrogram to identify the optimum number of clusters. The distance between X

and Y is more than A & B and P & Q, hence we choose that as the cut to create clusters and number of clusters chosen are 5
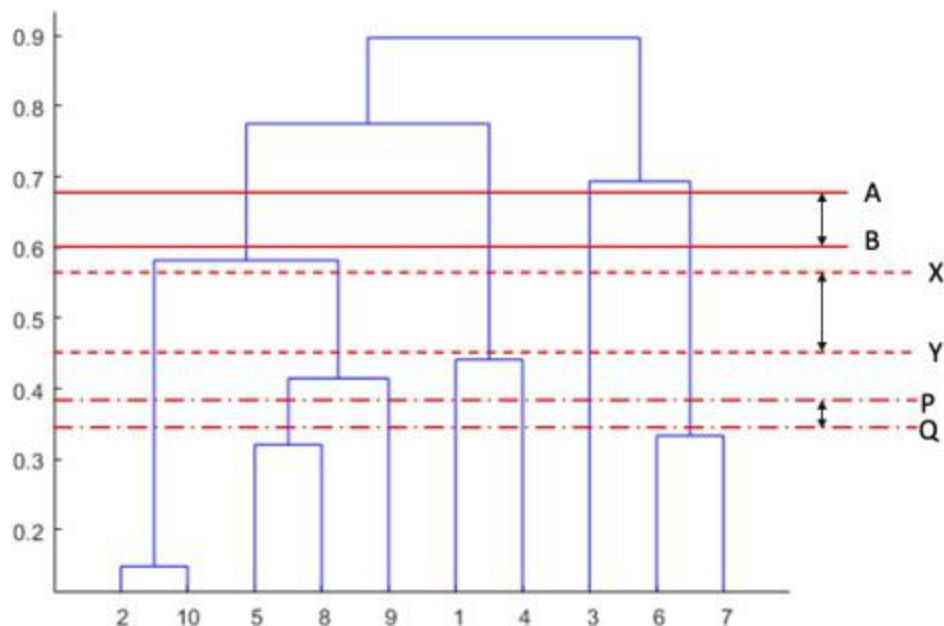
From a dendrogram, the number of clusters is given by the number of vertical lines being cut by a horizontal line. The *optimum* number of clusters is given by the number of the vertical lines in the dendrogram cut by a horizontal line such that it intersects the tallest of the vertical lines. Or if the cut is shifted from one end of the vertical line to another, the length hence covered is the maximum.

In the example shown in (Figure 2-13), we have shown three potential cuts – AB, PQ and XY. If we take a cut above AB it will result into two very broad clusters while below PQ will result in nine clusters which will become difficult to analyse further.

Here the distance between X and Y is more than A & B and P & Q. So we can conclude that the distance between X and Y is the maximum and hence we can finalise that as the best cut. This cut, intersects at five distinct points hence we should have five clusters. Hence, the height of the cut in the dendrogram is similar to the value of k in k-means clustering. In k-means clustering, "k" determines the number of clusters. In hierarchical clustering, the best cut determines the number of clusters we wish to have.

Similar to k-means clustering, the final number of clusters is not dependent on the choice from the algorithm only. The business acumen and the pragmatic logic plays a vital role in determining the final number of clusters. Recall that one of the important attribute of clusters is their usability which we discussed in section 1.2 earlier.

🎯 POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

1. What is the greedy approach used in hierarchical clustering?

2. Complete linkage is used for finding distances for closest neighbours – TRUE or FALSE

3. What is the difference between group linkage and ward linkage?

Describe the process to find the most optimal value of "k"

We have now covered the background of hierarchical clustering and how we determine the clusters. We will now discuss the advantages and challenges we face with hierarchical clustering.

2.4.4 Pros and cons of hierarchical clustering

Hierarchical clustering is a strong clustering technique and quite popular too. Similar to k-means, it is also using distance as a metric to measure the similarity. At the same time, there are a few challenges with the algorithm. We are discussing pros and cons of hierarchical clustering now. The advantages of hierarchical clustering are:

1. Perhaps the biggest advantage we have with hierarchical clustering is reproducibility of results. Recall in k-means clustering, the process starts with random initialization of centroids giving different results. In hierarchical clustering we can reproduce the results.
2. In hierarchical clustering, we do not have to input the number of clusters to segment the data.

3. The implementation is easy to implement and comprehend. Since it follows a tree-like structure, it is explainable to users from non-technical backgrounds.
4. The dendrogram generated can be interpreted to generate a very good understanding of the data with a visualization.

At the same time, we do face some challenges with hierarchical clustering algorithm which are :

1. The biggest challenge we face with hierarchical clustering is the time taken to converge. The time complexity for k-means is linear while for hierarchical clustering is quadratic. For example, if we have "n" data points, then for k-means clustering the time complexity will be O(n) while for hierarchical clustering is O(n).
2. Since the time complexity is O(n), it is a complex task. Hence the memory required to compute is at least $O(n^2)$ making hierarchical clustering quite a time consuming and memory intensive process. And this is the issue even if the dataset is medium. The computation required might not be a challenge if we are using really high-end processors but surely can be a concern for regular computers we use.
3. The interpretation of dendrograms at times can be subjective hence due diligence is required while interpretation of dendrograms. It can be subjective as different analysts can decipher different cuts and try to prove their methodology. Hence, it is advisable to interpret the results in the light of mathematics and marry the results with real-world business problem.
4. The hierarchical clustering cannot undo the previous steps it has done. In case, we feel that a connection made is not proper and should be rolled back, still there is no mechanism to remove the connection.
5. The algorithm is very sensitive to outliers and messy dataset. Presence of outliers, NULL, missing values, duplicates etc. make a dataset messy. And hence the resultant output might not be proper and not what we expected.

But despite all the challenges, hierarchical clustering is one of the most widely used clustering algorithms. Generally, we create both k-means clustering and hierarchical clustering for the same dataset to compare the

results of the two. If the number of clusters suggested and the distribution of respective clusters look similar, we get more confident on the clustering methodology used.

This marks our end of the theoretical and mathematical study of hierarchical clustering. It is time for action and jump into Python for coding.

2.4.5 Hierarchical clustering case study using Python

We will now create a Python solution for hierarchical clustering, using the same dataset we used for k-means clustering.

Step 1: Load the libraries and dataset till step 6 in k-means algorithm.

Step 7: Next, we are going to create hierarchical clustering using three linkages methods – average, ward and complete. Then the clusters are getting plotted. The input to the method is the X_Standard variable, linkage method used and the distance metric. Then, using matplotlib library, we are plotting the dendrogram. In the code snippet, simply change the method from 'average' to 'ward' and 'complete' and get the respective results.

```
from scipy.cluster.hierarchy import dendrogram, linkage
Z_df_average = linkage(X_standard, 'average', metric='euclidean')
Z_df_average.shape
plt.figure(figsize=(30, 12))
dendrogram(Z_df_average)
plt.show()
```
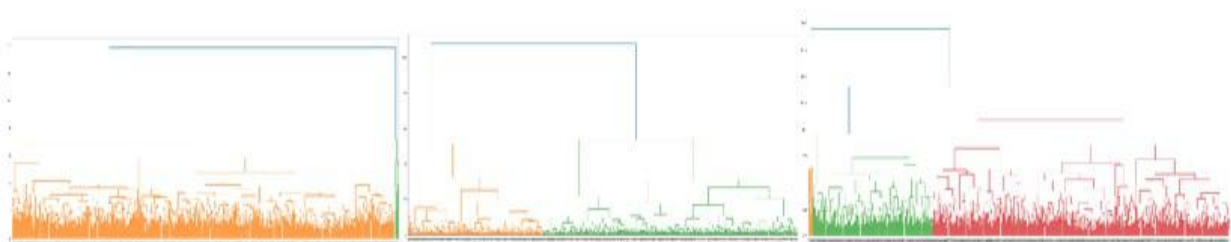

Step 8: We now want to choose the number of clusters we wish to have. For this purpose, let's recreate the dendrogram by sub-setting the last 10 merged clusters. We have chosen 10 as it is generally an optimal choice, you are advised to test with other values too.

```
dendrogram(
    Z_df_complete,
    truncate_mode='lastp',     p=10,)
plt.show()
```
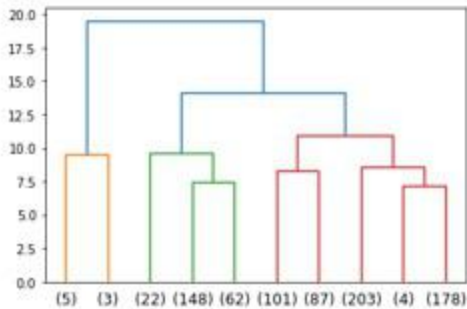

Step 9: We can observe that the most optimal distance is 10.

Step 10: Cluster the data into different groups. By using the logic described in the last section, the number of optimal clusters is coming to be four.

```
from scipy.cluster.hierarchy import fcluster
hier_clusters = fcluster(Z_df_complete, max_distance, criterion='
hier_clusters
len(set(hier_clusters))
```

Step 11: Plot the distinct clusters using matplotlib library.

```
plt.scatter(X_standard[:,0], X_standard[:,1], c=hier_clusters)
plt.show()
```
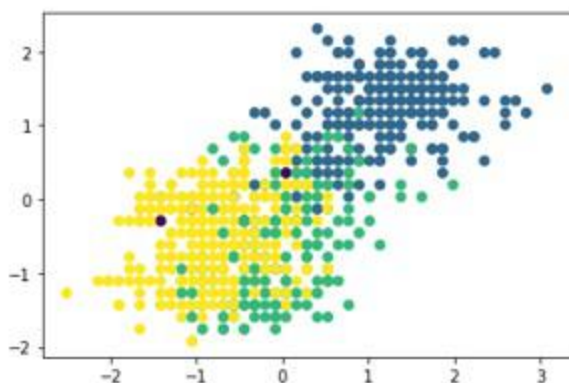

Step 12: For different values of distance, the number of clusters will change and hence the plot will look different. We are showing different results for

distances of 5, 15 20 and different number of clusters generated for each iteration. Here, we can observe that we get completely different results for different values of distances while we move from left to the right. We have to be cautious while we choose value of the distance and sometimes, we might have to iterate a few times to get the best value.
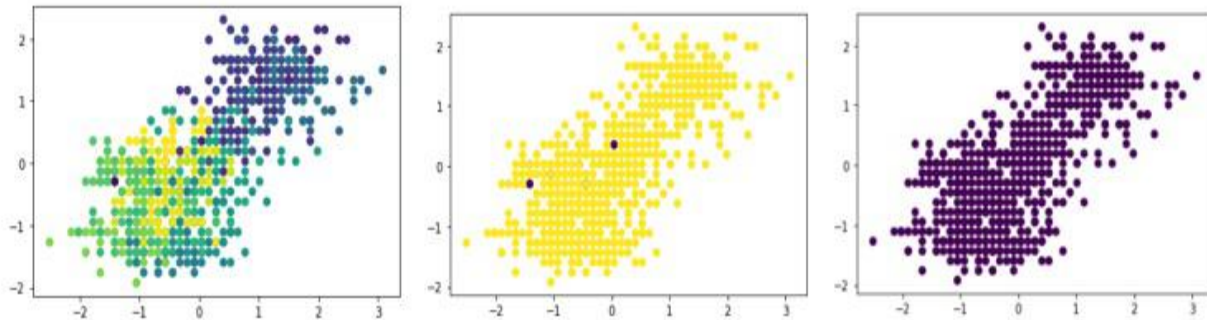
Hence, we can observe that using hierarchical clustering, we have segmented the data on the left-side to the one on the right-side of the Figure 2- below. The left side is the raw data while on the right we have representation of the clustered dataset.
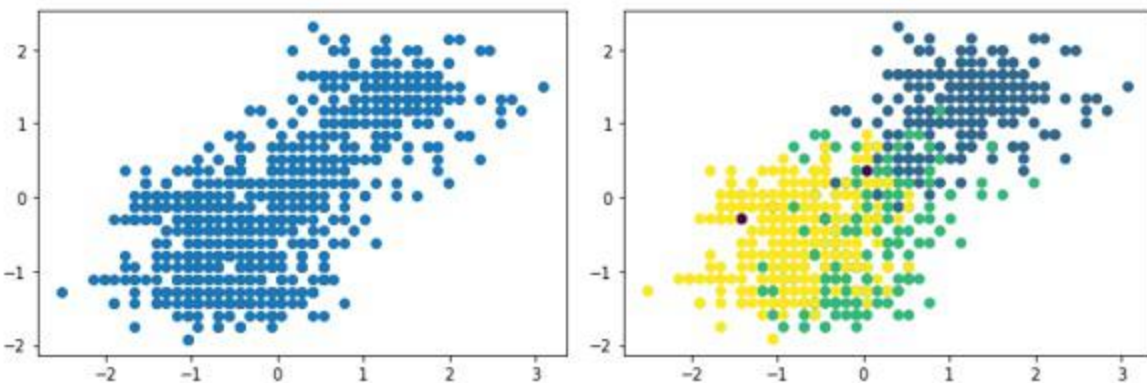
Hierarchical clustering is a robust method and is a highly recommended one. Along with k-means, it creates a great foundation for clustering-based solutions. Most of the time, at least these two techniques are worked upon when we create clustering solutions. And then we move to iterate with other methodologies.

This marks the end of discussion on connectivity-based clustering algorithms. We will now move forward to density-based solutions and discuss DBSCAN

clustering in the next section

2.5 Density based clustering

We have studied k-means in the earlier sections. Recall how it uses a centroid based method to assign a cluster to each of the data points. Even if an observation is an outlier, the outlier point pulls the centroid towards itself and is also assigned a cluster like a normal observation. These outliers do not necessarily bring information to the cluster and can impact other data points but are still made a part of the cluster. Moreover, getting clusters of arbitrary shape as shown in (Figure 2-14) is a challenge with k-means algorithm. Density based clustering methods solve the problem for us.

Figure 2- 14 DBSCAN is highly-recommended for irregular shaped clusters. With k-means we generally get spherical clusters, DBSCAN can resolve it for us
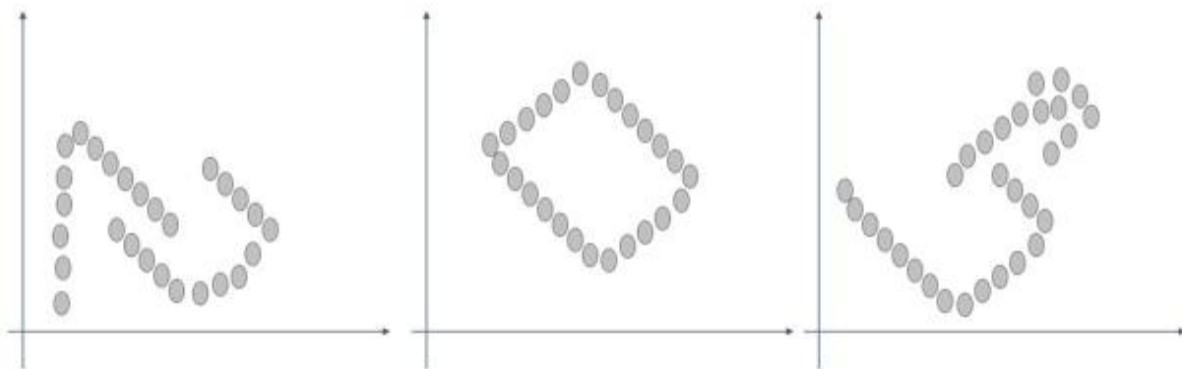
In a density-based clustering algorithm, we resolve both of these problems. In the density-based method, the clusters are identified as the areas which have higher density as compared to the rest of the dataset. In other words, given a vector-space diagram where the data points are represented – a cluster is defined by adjacent regions or neighbouring regions of high-density points. This cluster will be separated from other clusters by regions of low-density points. The observations in the sparse areas or separating regions are considered as noise or outliers in the dataset. The density-based clustering is shown in (Figure 2-).

We mentioned two terms - "neighbourhood" and "density". To understand density-based clustering, we will study these terms in the next section.

2.5.1 Neighborhood and density

Imagine we represent data observations in a vector-space. And we have a point P. We now define the neighbourhood for this point P. The representation is shown in (Figure 2-15) below.

Figure 2- 15 Representation of data points in a vector-space diagram. On the right-side we have a point P and the circle drawn is of radius ε. So, for ε > 0, the neighborhood of P is defined by the set of points which are at ε distance from the point P
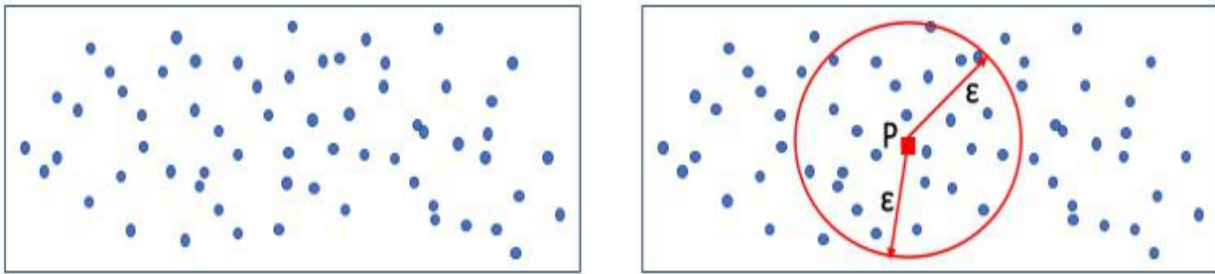
As we can make out from (Figure 2-15) above, for a point P we have defined a ε - neighborhoods for it which are the points equidistant from P. In a 2-D space, it is represented by a circle, in a 3-D space it is sphere and for a n-dimensional space it is n-sphere with center P and radius ε. This defines the concept of *neighborhood*.

Now, let's explore the term, *density*. Recall density is mass divided by volume (mass/volume). Higher the mass, higher the density and lower the mass, lower the density. Conversely, lower the volume, higher the density and higher the volume, lower the density.

In the context above, mass is the number of points in the neighborhood. In (Figure 2-16) below, we can observe the impact of ε on the number of data points or the mass.

Figure 2-16 The impact of radius ε, on the left side the number of points is more than on the right-side. So, the mass of right side is less since it contains a smaller number of data points
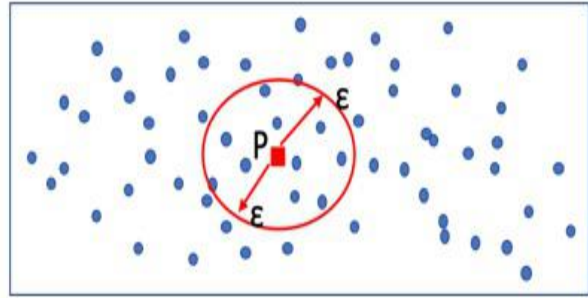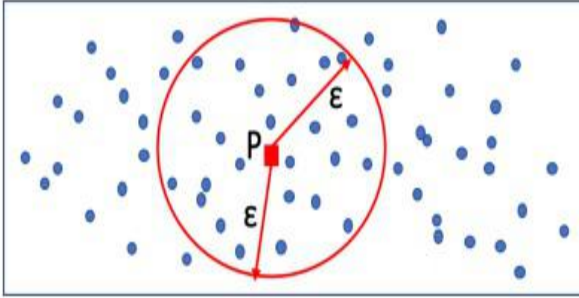
When it comes to volume, in the case of a 2-d space, volume is πr^2, while for a sphere which is three-dimensional it is $4/3\ \pi r^3$. For spheres of n-dimensions, we can calculate the respective volume as per the number of dimensions.

So, in the two cases shown in (Figure 2-16), for a point "P" we can get the number of points (mass) and volumes and then we can calculate the respective densities. But the absolute values of these densities mean nothing to us. It is used to cluster the points having similar density. Or in other words, the points which are in the same neighbourhood and have similar densities can be clubbed in one cluster.

In an ideal case scenario, we wish to have highly dense clusters having maximum number of points. In the two cases shown in (Figure 2-17) below, we have a less dense cluster depicted on the left and high-dense one on the right-hand side.

Figure 2-17 Denser clusters are preferred over less dense ones. Ideally a dense cluster, with maximum number of data points is what we aim to achieve from clustering
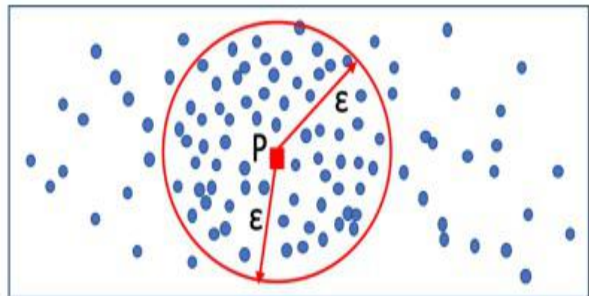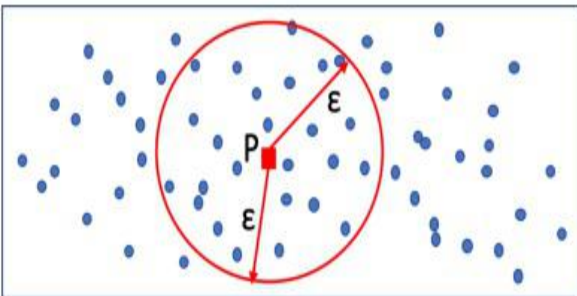
From the discussion above, we can conclude that:

1. If we *increase* the value of ε, we will get a *higher* volume but not

necessarily a *higher* number of points (mass). It depends on the distribution of the data points.

2. Similarly, if we *decrease* the value of ε, we will get *lower* volume but not necessarily *lower* number of points (mass).

It is the fundamental principle we adhere to. Hence, it is imperative that while choosing the clusters we choose clusters which have high density and cover the maximum number of neighboring points.

We have hence concluded the concepts for density-based clustering. These concepts are the building blocks for DBSCAN clustering which we are discussing next!

2.5.2 DBSCAN Clustering

Density-Based Spatial Clustering of Applications with Noise or DBSCAN clustering is a one of the highly recommended density-based algorithms. It clusters the data observations which are closely packed in a densely populated area but not considering the outliers in low-density regions. Unlike k-means, we do not specify the number of clusters and the algorithm is able to identify irregular shaped clusters whereas k-means generally proposes spherical-shaped clusters. Similar to hierarchical clustering, it works by connecting the data points but with the observations which satisfy the density-criteria or the threshold value.

Note:

DBSCAN was proposed in 1996 by Martin Ester, Hans-Peter Kriegal, Jörg Sander and Xiaowei Xu. The algorithm was awarded the test of time award in 2014 at ACM SIGKDD. The paper can be assessed at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.1980.

DBSCAN works on the concepts of neighbourhood we discussed in the last section. We will now dive deep into the working methodology and building blocks of DBSCAN.

nuts and bolts of DBSCAN clustering

We will now examine the core building blocks of DBSCAN clustering. We know it is a density-based clustering algorithm and hence neighbourhood concept is applicable over here.

Consider we have a few data observations which we need to cluster. We also locate a data point "P". Then, we can easily define two terms:

1. The radius of the neighbourhood around P, called as ε, which we have discussed in the last section.
2. The minimum number of points we wish to have in the neighbourhood of P or in other words, minimum number of points that are required to create a dense region. This is referred to as *minPts*. And is one of the parameters we can input by applying a threshold on minPts.

Based on the concepts above, we can classify the observations in three broad categories - core points, border or reachable points and outliers:

3. **Core points**: any data point "x" can be termed as a core point if at least minPts are within ε distance of it (including x), shown as squares in (Figure 2-18) below. They are the building blocks of our clusters and hence are called core. We use the same value of radius (ε) for each point and hence the *volume* of each neighborhood remains constant. But the number of points will vary and hence the *mass* varies. And hence, the density varies as well. Since we put a threshold using minPoints, we are putting a limit on density. So, we can conclude that core points fulfil the minimum density threshold requirement. It is imperative to note that minPts can be used to iterate and fine-tune the clusters.

Figure 2-18 Core points are shown in square, border points are shown in filled circle while noise is unfilled circles. Together these three are the building blocks for DBSCAN clustering
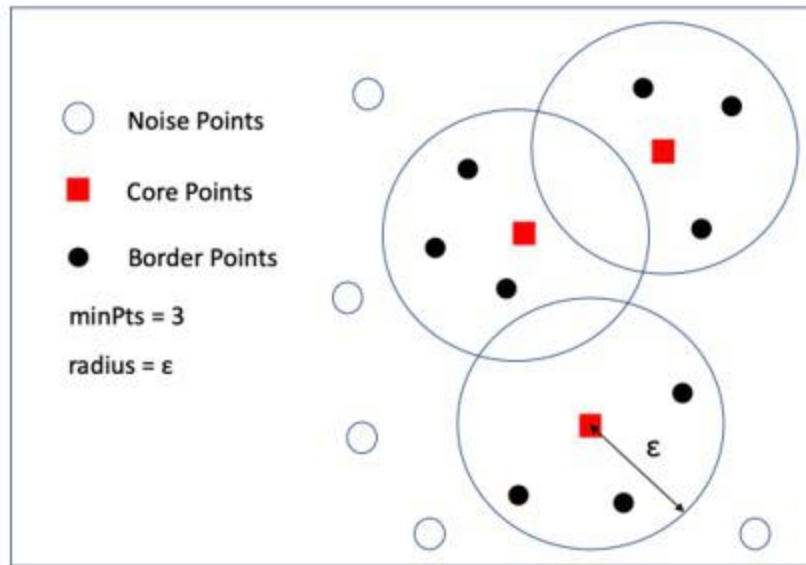
4. **Border points or reachable points**: a point which is not a core-point in the clusters is called a border point, shown as filled circles in (Figure 2-18) above.

A point "y" is directly reachable from x if y is within ε distance of core point x. A point can only be approached from a core point and it is the primary condition or rule to be followed. Only a core-point can reach a non-core point and the opposite is not true. In other words, a non-core point can only be reached, it cannot reach anyone else.

To understand the process better, we have to understand the term *density-reachable* or *connectedness*. As shown in (Figure 2-19) below, we have two core points X and Y. We can directly go from X to Y. Point Z is not in the neighbourhood of X but is the neighbourhood of Y. So, we cannot directly reach Z from X. But we can surely reach Z from X through Y or in other words using neighbourhood of Y, we can travel to Z from X.

Figure 2-19 X and Y are the core points and we can travel from X to Y. Though Z is not in the immediate neighbourhood of X, we can still reach Z from X through Y. It is the core concept of density-connected points

Core Points
Border Points
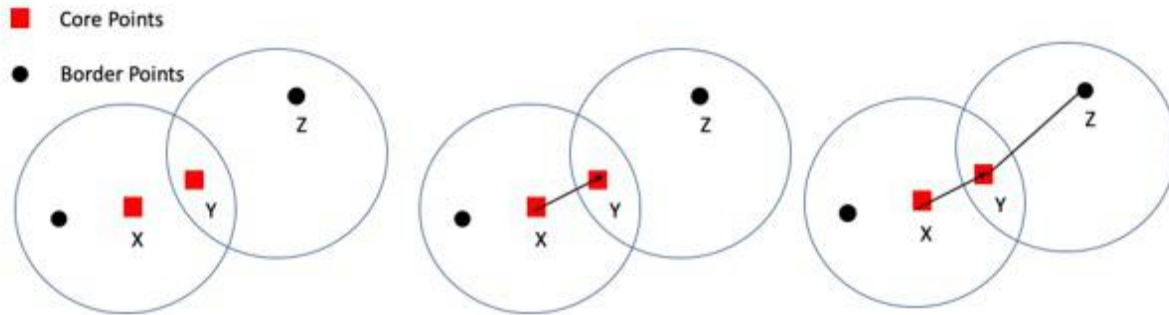
5. **Outliers**: all the other points are outliers. In other words, if it is not a core point or is not a reachable point, it is an outlier, shown as unfilled circles in (Figure 2-18) above. They are not assigned any cluster.

Now we have defined the building block for DBSCAN. We will now proceed to the process followed in DBSCAN in the next section.

steps in DBSCAN clustering

Now we have defined the building block for DBSCAN. We will now examine the steps followed in DBSCAN:

1. We start with assigning values for ε and minimum points (minPts) required to create a cluster.
2. We start with picking a random point let's say "P" which is not yet given any label i.e. which has not been analysed and assigned any cluster.
3. We then analyse the neighbourhood for P. If it contains sufficient number of points i.e. higher than minPts, then the condition is met to start a cluster. We can call the point P as *core-point*. If a point cannot be recognised as a core-point, we will assign it the tag of *outlier* or *noise*. We should note this point can be made a part of a different cluster later.
4. Once this core point is finalised, we start creating this cluster by adding all directly reachable points from P and then increase this cluster size by adding more directly reachable from P. Then we add all the points to the cluster, which can be included using the neighbourhood. If we add an outlier point to the cluster, the tag of the outlier point is changed to border point.
5. This process continues till density-cluster is complete. We then find a

new unassigned point and repeat the process.
6. Once all the points have been assigned to a cluster or called as an outlier, we stop our clustering process.

There are iterations done in the process. And once the clustering concludes, we utilize business logic to either merge or split a few clusters.

🎯 POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

1. Compare and contrast the importance of DBSCAN clustering with respect to kmeans clustering.

2. A non-core point can reach a core point and vice versa is also true – TRUE or FALSE

3. Explain the significance of neighbourhood and minPts.

Describe the process to find the most optimal value of "k"

Now we are clear with the process of DBSCAN clustering. Before creating the Python solution, we will examine advantages and disadvantages with the DBSCAN algorithm.

pros and cons of DBSCAN clustering

DBSCAN has following advantages:

1. Unlike k-means, we need not specify the number of clusters to DBSCAN.
2. The algorithm is quite a robust solution for unclean datasets. Unlike other algorithms, it can deal with outliers effectively.
3. We can determine irregular shaped clusters too. Perhaps, it is the biggest advantage with DBSCAN clustering.
4. Only the input of radius and minPts is required by the algorithm.

DBSCAN suffers from following challenges:

1. The differentiation in clusters is sometimes not clear using DBSCAN. Depending on the order of processing the observations, a point can change its cluster. In other words, if a border point P is accessible by more than one cluster, P can belong to either cluster, which is dependent on the order of processing the data.
2. If the difference in densities is very big, then the optimum combination of ε and minPts will be difficult to determine and hence, DBSCAN will not be generating effective results.
3. The distance metric used plays a highly significant role in DBSCAN. Arguably, the most common metric used in Euclidean distance, but if the number of dimensions are quite large then it becomes a challenge to compute.
4. The algorithm is very sensitive to different values of ε and minPts. Sometimes, getting the most optimum value becomes a challenge.

We will now create a Python solution for DBSCAN clustering.

python solution for DBSCAN clustering

We will use the same dataset we have used for k-means and hierarchical clustering.

Step 1: Load the libraries and dataset till step 6 in k-means algorithm.

Step 7: Import additional libraries

```
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from sklearn.neighbors import NearestNeighbors
```

Step 8: We are fitting the model with a value for minDist and radius.

```
db_default = DBSCAN(eps = 0.0375, min_samples = 6).fit(X_standard
labels = db_default.labels_
```

Step 9: The number of distinct clusters are one.

```
list(set(labels))
```

Step 10: We are not getting any results for clustering here. In other words, there will not be any logical results of clustering since we have not provided the optimal values for minPts and ε. So, first we will find out the optimum values for the ε. For this, we will calculate the distance to nearest points for each point and then sort and plot the results. Wherever the curvature is maximum, it is the best value for ε. For minPts, generally minPts ≥ d+1 where d is the number of dimensions in the dataset.

Note:

You are advised to go through the paper at the link to further study on how to choose the values of radius for DBSCAN
https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf

```
neigh = NearestNeighbors(n_neighbors=2)
nbrs = neigh.fit(X_standard)
distances, indices = nbrs.kneighbors(X_standard)
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
```
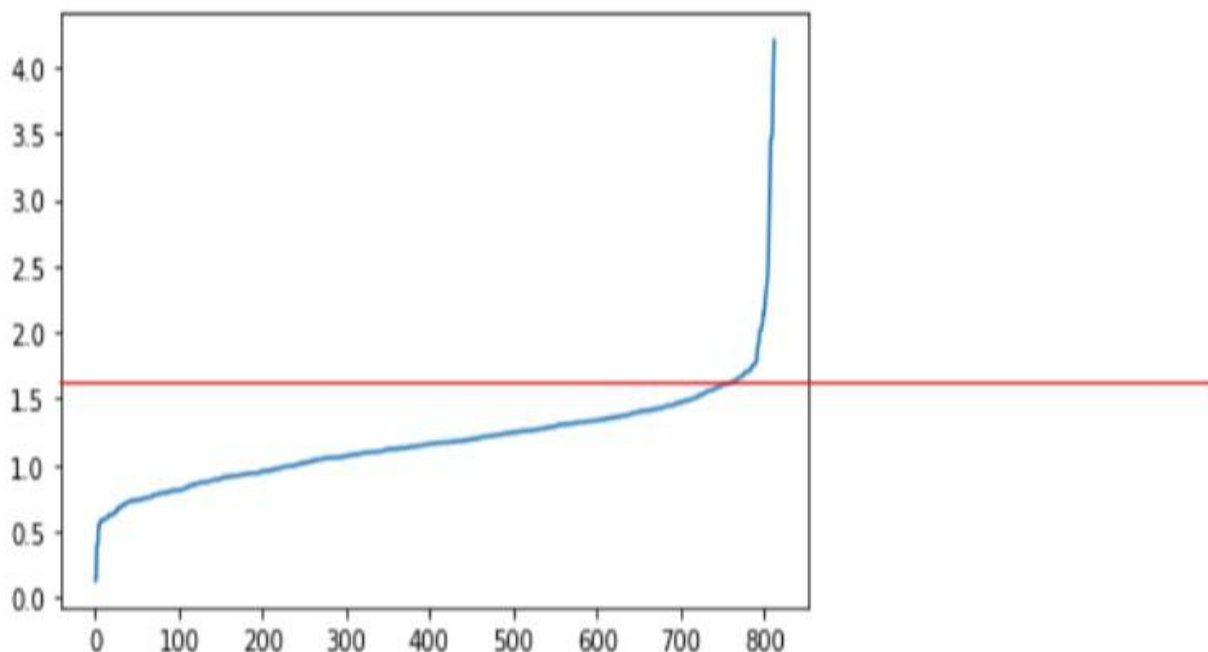

Step 10: The best value is coming as 1.5 as observed the point of defection

above. We will use it and set the minPts as 5 which is generally taken as a standard.

```
db_default = DBSCAN(eps=1.5, min_samples=5)
db_default.fit(X_standard)
clusters = db_default.labels_
```

Step 11: Now we can observe that we are getting more than one cluster.

```
list(set(clusters))
```

Step 12: Let's plot the clusters.

```
colors = ['blue', 'red', 'orange', 'green', 'purple', 'black', 'b
vectorizer = np.vectorize(lambda x: colors[x % len(colors)])
plt.scatter(X_standard[:,0], X_standard[:,1], c=vectorizer(cluste
```
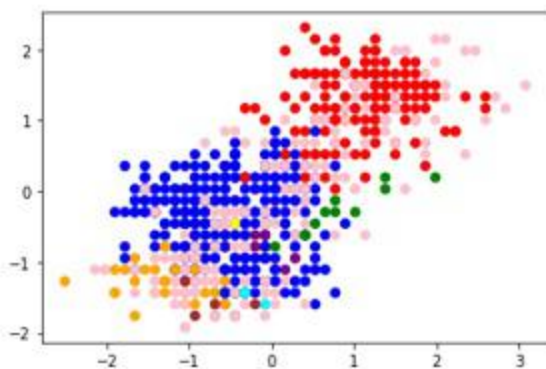

We have thus created the solution using DBSCAN. You are advised to compare the results from all the three algorithms. In real-world scenario, we test the solution with multiple algorithms, iterate with hyperparameters and then choose the best solution.

Density-based clustering is quite a novel solution and to a certain extent very effective one too. It is heavily recommended if the shape of the clusters is uneven. We have studied DBSCAN now, there is one more algorithm called OPTICS which we will study in subsequent chapters in the book.

With this, we conclude our discussion on DBSCAN clustering. In the next section, we are solving a business use case on clustering. In the case study, the focus is less on technical concepts but more on the business

understanding and the solution generation.

2.6 Case study using clustering

We will now define a case study which employs clustering as one of the solutions. This case study-based approach is generally followed in job related interviews wherein a case is discussed during the interview stage. And hence, it is highly recommended for you to understand how we implement machine learning solutions in pragmatic business scenarios.

A case study typically has a business problem, the data set available, the various solutions which can be used, challenges faced and the final chosen solution. We also discuss the issues faced while implementing the solution in real-world business.

So, let's start our case study on clustering using unsupervised learning. In the case study, we are focussing on the steps we take to solve the case study and not on the technical algorithms, as there can be multiple technical solutions to a particular problem.

Business context: The industry we are considering can be retail, telecom, BFSI, aviation, health-care. Basically, any business which deals with customers (almost all businesses have customers). For any business, one of the many objectives can be to increase the revenue/sales. And to increase the revenue, the business would wish to have increasingly newer customers. The business would also wish the existing consumers to buy more and buy more often. So, the business always strives to keep the consumers engaged, keep them happy and increase their transactional value with themselves.

For this to happen, the business should have a thorough understanding of a consumer base, know their preferences, tastes, price points, liking of categories etc. And once the business has examined and understood the consumer base minutely, then:

1. The product team can improve the product features as per the consumer's need.
2. The pricing team can improve the price of the products by aligning them

to customer's preferred prices. The prices can be customized for a customer or loyalty discounts can be offered.
3. The marketing team and customer relationship team (CRM), can target the consumers by a customized offer.
4. The teams can win-back the consumers which are going to churn or stop buying from the business, can enhance their spend, increase the stickiness and increase the customer lifetime value.
5. Overall, different teams can align their offerings as per the understanding of the consumers generated. And the end consumer will be happier, more engaged, more loyal to the business leading to more fruitful consumer engagement.

The business hence has to dive deep into the consumers data and generate understanding of the base. The customer data can look like as shown in the next section.

Dataset for the analysis: We are taking an example for an apparel retailer (H&M, Uniqlo etc). A retailer having a loyalty program saves the customer's transaction details. The various (not exhaustive) data sources can be as shown below:

CustID	Revenue	Invoices	Items	Discount%
123	100	2	10	30
124	101	3	12	0
125	102	4	15	5
126	103	2	11	40

CustID	DOB	City	Gender
123	01/01/1990	A	M
124	02/01/1990	B	F
125	03/01/1990	C	M
126	04/01/1990	D	F

ItemNo	Price	Sub-category	Category
1	10	A	P
2	11	B	Q
3	12	C	R
4	10	D	S

StoreID	StoreName	City	Area
100	XYZ	A	1000
101	PQR	B	2000
102	ABC	C	1500
103	TUV	D	2500

We can have store details which have all the details of a store like store ID, store name, city, area, number of employees etc. We can have the item hierarchies table which has all the details of the items like price, category etc. Then we can have customer demographic details like age, gender, city and customer transactional history which has details of the consumer's past sales with us. Clearly, by joining such tables, we will be able to create a master

table which will have all the details at one place.

Note:

You are advised to develop a good skill set for SQL. It is required in almost each of the domains related to data – be it data science, data engineering or data visualization, SQL is ubiquitous.

We are depicting an example of master table below. It is not an exhaustive list of variables and number of variables can be much larger than the ones below. The master table has some raw variables like Revenue, Invoices etc. and derived variables like Average Transaction value and Average basket size etc.

CustID	Revenue	Invoices	ItemsBought	Age	Gender	Avg Txn Value	Avg Basket Size	DaysSinceLastTxn	City
1	1000	2	2	25	M	500	1	20	A
2	2000	4	5	26	F	500	1.25	12	B
3	3000	3	4	27	M	1000	1.33	30	C
4	4000	4	5	28	F	1000	1.25	25	D
5	5000	2	1	29	F	2500	0.5	1	E

We could also take an example of a telecom operator. In that subscriber's usage, call rate, revenue, days spent on network, data usage etc. will be the attributes we will be analysing. Hence, based on the business domain at hand, the data sets will change.

Once we have got the data set, we generally create derived attributes from them. For example, the average transaction value attributes is total revenue divided by the number of invoices. We create such attributes in addition to the raw variables we already have.

Suggested solutions: There can be multiple solutions to the problem, some of which we are describing below:

1. We can create a dashboard to depict the major KPI (key performance indicators). It will allow us to analyse the history and take necessary actions based on it. But the solution will be more reporting in nature with trends, KPI which we already are aware of.
2. We can perform data analysis using some of techniques we used in the

solutions in the earlier sections. It will solve a part of the problem and moreover, it is difficult to consider multiple dimensions simultaneously.

3. We can create predictive models to predict if the customers are going to shop in the coming months or going to churn in the next X days, but it will not solve the problem completely. To be clear, churn here refers that customer no longer shops with the retailer in the next X days. Here, duration X is defined as per the business domain. For example, for telecom domain X will be lesser than insurance domain. It is due to the fact that people use mobile phone everyday whereas for insurance domain, most customers might be paying premium yearly. So customer interaction is less for insurance.

4. We can create customer segmentation solutions wherein we are grouping customers based on their historical trends and attributes. This is the solution we will use to solve this business problem.

Solution for the problem: Recall in Chapter 1 in (Figure 1-9), where we discussed the steps, we follow in the machine learning algorithm. Everything starts with defining the business problem and then data discovery, pre-processing etc. For the case study above, we will utilize the similar strategy. We have already defined the business problem; data discovery is done and we have completed the EDA and pre-processing of the data. We wish to create the segmentation solution using clustering.

Step 1: We start with finalizing the dataset we wish to feed to the clustering algorithms. We might have created some derived variables, treated some missing values or outliers etc. In the case study, we would want to know the minimum/maximum/average values of transactions, invoices, items bought etc. We would be interested to know the gender and age distribution. We also would like to know the mutual relationships between these variables like if women customers use online mode more than male customers. All these questions are answered a part of this step.

A Python Jupyter notebook is checked-in at the Github repository, which provides detailed steps and code for the exploratory data analysis(EDA) and data pre-processing. Check it out!

Step 2: We create the first solution using k-means clustering followed by hierarchical clustering. For each of the algorithms, iterations are done by

changing hyperparameters. In the case study, we will choose parameters like number of visits, total revenue, distinct categories purchased, online/offline transactions ratio, gender, age etc. as parameters for clustering.

Step 3: A final version of the algorithm and respective hyperparameters are chosen. The clusters are analysed further in the light of business understanding.

Step 4: More often, the clusters are merged or broken, depending on the size of the observations and the nature of the attributes present in them. For example, if the total customer base is 1 million, it will be really to action on cluster of size 100 . At the same time, it will be equally difficult to manage a cluster of size 700,000.

Step 5: We then analyse the clusters we have finally got. The clusters distribution is checked for the variables, their distinguishing factors are understood and we give logical names to the clusters. We can expect to see such a clustering output as shown in (Figure 3-) below.

In the example clusters shown below, we have depicted spending patterns, responsiveness to previous campaigns, life stage, overall engagement as a few dimensions. And respective sub-divisions of each of these dimensions has also been shown. The clusters will be a logical combination of these dimensions. The actual dimensions can be much higher.

The segmentation shown above can be used for multiple domains and businesses. The parameters and attributes might change, the business context is different, the extent of data available might vary – but the overall approach remains similar.

In addition to the few applications, we saw in the last section, we are examining some of the use cases now:

1. Market research utilizes clustering to segment the groups of consumers into market segments. And then the groups can be analysed better in terms of their preferences. The product placement can be improved, pricing can be made tighter and geography selection will be more scientific.
2. In the bioinformatics and medical industry, clustering can be used to group the genes into distinct categories. Groups of genes can be segmented and comparisons can be assessed by analysing the attributes of the groups.
3. It is used as an effective data pre-processing step before we create algorithms using supervised learning solutions. It can also be used to reduce the data size by focussing on the data points belonging to a cluster.
4. It is utilized for pattern detection across both structured and unstructured dataset. We have already studied the case for structured dataset. For text data, it can be used to group similar types of documents, journals, news etc. We can also employ clustering to work and develop solutions for images. We are going to study unsupervised learning solutions for text and images in later chapters.
5. As the algorithms work on similarity measurements, it can be used to segment the incoming data set as fraud or genuine, which can be used to reduce the amount of criminal activities.

The use cases of clustering are quite a lot. We have discussed only the prominent ones. It is one of the algorithms which change the working methodologies and generate a lot of insights around the data. It is widely used across telecom, retail, BFSI, aviation etc.

At the same time, there are a few issues with the algorithm. We will now examine the common problems we face with clustering in the next section.

2.7 Common challenges faced in clustering

Clustering is not a completely straight-forward solution without any challenges. Similar to any other solution in the world, clustering too has its share of issues faced. We are discussing the most common challenges we face in clustering which are:

1. Sometimes the magnitude of the data is quite big and there are a lot of dimensions available. In such a case, it becomes really difficult to manage the data set. The computation power might be limited and like any project, there is finite time available. To overcome the issue, we can:
2. Try to reduce the number of dimensions by finding the most significant variables by using supervised learning-based regression approach or decision tree algorithm etc.
3. Reduce the number of dimensions by employing Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) etc.
4. Noisy data set: "Garbage in garbage out" – this cliché is true for clustering too. If the data set is messy it creates a lot of problems. The issues can be:
5. Missing values i.e. NULL, NA, ? , blanks etc.
6. Outliers are present in the dataset.
7. Junk values are present like #€¶§^ etc. are present in the dataset.
8. Wrong entries are made in the data. For example, if name are entered in the revenue field it is an incorrect entry.

We are going to discuss the steps and process to resolve these issues in each of the chapters. In this chapter, we are examining – how to work with categorical variables

3. Categorical variables: Recall that while discussing we discussed the issue with k-means not able to use categorical variables. We are solving that issue now.

To convert categorical variables into numeric one, we can use *one-hot encoding*. This technique adds additional columns equal to the number of distinct classes as shown in (Figure 2-) below. The variable city has unique

values as London and NewDelhi. We can observe that two additional columns have been created with 0 or 1 filled for the values.

CustID	City	Sales	Age		CustID	London	NewDelhi	Sales	Age
1234	London	100	25		1234	1	0	100	25
1235	NewDelhi	101	26		1235	0	1	101	26
1236	NewDelhi	102	27		1236	0	1	102	27
1237	NewDelhi	103	28		1237	0	1	103	28
1238	London	104	29		1238	1	0	104	29

But using one-hot encoding does not always ensure an effective and efficient solution. Imagine if the number of cities in the example above are 100, then we will have 100 additional columns in the dataset and most of the values will be filled with zero. Hence, in such a situation it is advisable to group few values.

4. Distance metrics: with different distance metrics we might get different results. Though there is no "one size fits all", mostly you would find Euclidean distance being used for measuring distance.
5. Interpretations for the clusters are quite subjective. By using different attributes, completely different clustering can be done for the same datasets. As discussed earlier, the focus should be on solving the business problem at hand. This holds the key to choose the hyperparameters and the final algorithm.
6. Time consuming: since a lot of dimensions have to be dealt simultaneously, sometimes converging the algorithm takes a lot of time.

But despite all these challenges, clustering is a widely recognized and utilized technique. We have discussed the use cases of clustering in the real-world in the last section.

This marks the end of the discussion on challenges with clustering. You can proceed to summary now.

2.8 Summary

Unsupervised learning is not an easy task. But it is certainly a very engaging

one. It does not require any target variable and the solution identifies the patterns itself, which is one of the biggest advantages with unsupervised learning algorithms. And the implementations are already creating a great impact on the business world. We studied one of such solution classes called clustering in this chapter.

Clustering is an unsupervised learning solution which is useful for pattern identifications, exploratory analysis and of course segmenting the data points. Organizations heavily use clustering algorithms and proceed to the next level of understanding the consumer data. Better prices can be offered, more relevant offers can be suggested, consumer engagement can be improved and overall customer experience is fantastic. After all, a happy consumer is the goal of any business. Not only structured data, we can use clustering for text data, images, videos and audios too. Owing to its capability in finding patterns across multiple data sets using a large number of dimensions – clustering is the go-to solution whenever we want to analyse multiple dimensions together.

In this second chapter of this book, we introduced concepts of unsupervised based clustering methods. We examined different types of clustering algorithms – k-means clustering, hierarchical clustering and DBSCAN clustering along with their mathematical concepts, respective use cases, pros and cons with an emphasis on creating actual Python code for the same datasets.

In the following chapter, we will study dimensionality reduction techniques like PCA and SVD. The building blocks for techniques, their mathematical foundation, advantages and disadvantages, use cases and actual Python implementation will be done.

You can proceed to the question section now!

Practical next steps and suggested readings

1. Get the online retail data from the link (https://www.kaggle.com/hellbuoy/online-retail-customer-clustering). This dataset contains all the online trasactions occurring between

1/12/2010 and 09/12/2011 for a UK based retailer. Apply the three algorithms described in the chapter to identify which customers the company should target and why.

2. Get the IRIS dataset from the link (https://www.kaggle.com/uciml/iris). It includes three iris species with 50 samples each having some properties of the flowers. Use kmeans and DBSCAN and compare the results.

3. Explore the dataset at UCI for clustering (http://archive.ics.uci.edu/ml/index.php)

4. Study the following papers on kmeans clustering, hierarchical clustering and DBSCAN clustering

5. Kmeans algorithm:

6. i. https://www.ee.columbia.edu/~dpwe/papers/PhamDN05-kmeans.pdf

7. ii. https://www.researchgate.net/publication/271616608_A_Clustering_Met Means_Algorithm

8. iii. https://ieeexplore.ieee.org/document/1017616

9. Hierarchical clustering

10. i. https://ieeexplore.ieee.org/document/7100308

11. ii. https://papers.nips.cc/paper/7200-hierarchical-clustering-beyond-the-worst-case.pdf

12. iii. https://papers.nips.cc/paper/8964-foundations-of-comparison-based-hierarchical-clustering.pdf

13. DBSCAN clustering

14. i. https://arxiv.org/pdf/1810.13105.pdf

15. ii. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.9220

3 Dimensionality reduction

"Knowledge is a process of piling up facts; wisdom lies in their simplification." – Martin H. Fischer

We face complex situations in life. Life throws multiple options to us and we choose a few viable ones from them. This decision of shortlisting is based on the significance, feasibility, utility and perceived profit from each of the options. And the ones which fit the bill are then chosen. A perfect example can be selecting the holiday destinations. Based on the weather, travel time, safety, food, budget and a number of other options; we choose a few where we would like to spend the next holidays. In this chapter, we are studying precisely the same – how to reduce the number of options, albeit in the data science and machine learning world.

In the last chapter we covered major clustering algorithms. We also studied a case-study over there. The datasets we generate and use in such real-world examples have a lot of variables. Sometimes, there can be more than a 100 variables or *dimensions* in the data. But not all of them are important; not all of them are significant. This is referred to as "*Curse of Dimensionality*". To perform any further analysis we choose a few from the list of all of the dimensions or variables. In this chapter we are going to study the need for dimension reductions, various dimensionality techniques and the respective pros and cons. We will dive deeper into the concepts of Principal Component Analysis (PCA) and SVD (Singular Value Decomposition), their mathematical foundation and complement with the Python implementation. And continuing our structure from the last chapter, we will examine a real-world case study in the telecommunication sector towards the end. There are other advanced dimensionality reduction techniques like t-SNE, LDA which we will explore in later chapters.

Clustering and dimensionality reductions are the major categories of unsupervised learning. We studied major clustering methods in the last chapter and we will cover dimensionality reduction in this chapter. With these two solutions, we would cover a lot of ground in the unsupervised

learning domain. But there are much more advanced topics to be covered, which are part of the latter chapters of the book.

In this third chapter, we are going to cover the following topics:

1. Curse of dimensionality and its disadvantages
2. Various methods of reducing dimensions
3. Principal Component Analysis (PCA)
4. Singular Value Decomposition (SVD)
5. Python solutions for both PCA and SVD
6. Case study on dimension reduction

Let's first understand what we mean by *"Curse of dimensionality"*.

3.1 Technical toolkit

We are using the 3.6+ version of Python as used in the last chapters. Jupyter Notebook will be used in this chapter too.

All the datasets and code files are checked-in to the Github repository at (https://github.com/vverdhan/UnsupervisedLearningWithPython/tree/main/Ch You need to install the following Python libraries to execute the `numpy`, `pandas`, `matplotlib`, `scipy`, `sklearn`. Since you have used the same packages in the last chapter, you need not install them again. CPU is good enough for execution, but if you face some computing lags, switch to GPU or Google colab. You can refer to Appendix to the book if you face any issues in the installation of any of these packages.

Now, we are starting with *"Curse of dimensionality"* in the following section.

3.2 Curse of Dimensionality

Let us continue from the holiday destination example we introduced earlier. The choice of the destination is dependent on a number of parameters – safety, availability, food, night-life, weather, budget, health and so on. Having too many parameters to decide is a confusing thought. It is the *Curse of Dimensionality*. Let us understand by a real-life example.

Consider this. A retailer wishes to launch a new range of shoes in the market. And for that a target group of customers have to be chosen. This target group will be reached through email, newsletter etc. The business objective is to entice these customers to buy the newly launched shoes. From the entire customer base, the target group of customers can be chosen based on the variables like customer's age, gender, pocket-size, preferred category, average spend, frequency of shopping and so on. These many variables or *dimensions* give us a hard time to shortlist the customers based on a sound data analysis technique. We would be analysing too many parameters simultaneously, examining the impact of each on the shopping probability of the customer and hence it becomes too tedious and confusing a task. It is the *Curse of Dimensionality* problem we face in the real-world data science projects. We can face curse of dimensionality in one more situation wherein the number of observations are lesser than the number of variables. For example, consider a dataset where the number of observations are X while the number of variables are more than X – in such a case we face Curse of Dimensionality.

An easy method to understand any dataset is through visualization. Let's visualize a dataset in a vector-space diagram. If we have only one attribute or feature in the dataset, we can represent it in one dimension. It is shown in (Figure 3-1(i)) below. For example, we might wish to capture only the height of an object using a single dimension. In case we have two attributes, we need two dimensions as shown in (Figure 3-1(ii)), wherein to get the area of an object we will require both length and breadth. In case we have three attributes, for example to calculate the volume which requires length, breadth and height, it requires a three-dimensional space, as shown in (Figure 3-1(iii)). And this requirement will continue to grow based on the number of attributes.

Figure 3-1 (i) Only one dimension is required to represent the data points, for example to represent height of an object (ii) We need two dimensions to represent a data point. Each data point can correspond to length and breadth of an object which can be used to calculate the area (iii) Three dimensions are required to show a point. Here, it can be length, breadth and height which are required to get the volume of an object. This process continues based on the number of dimensions present in the data.
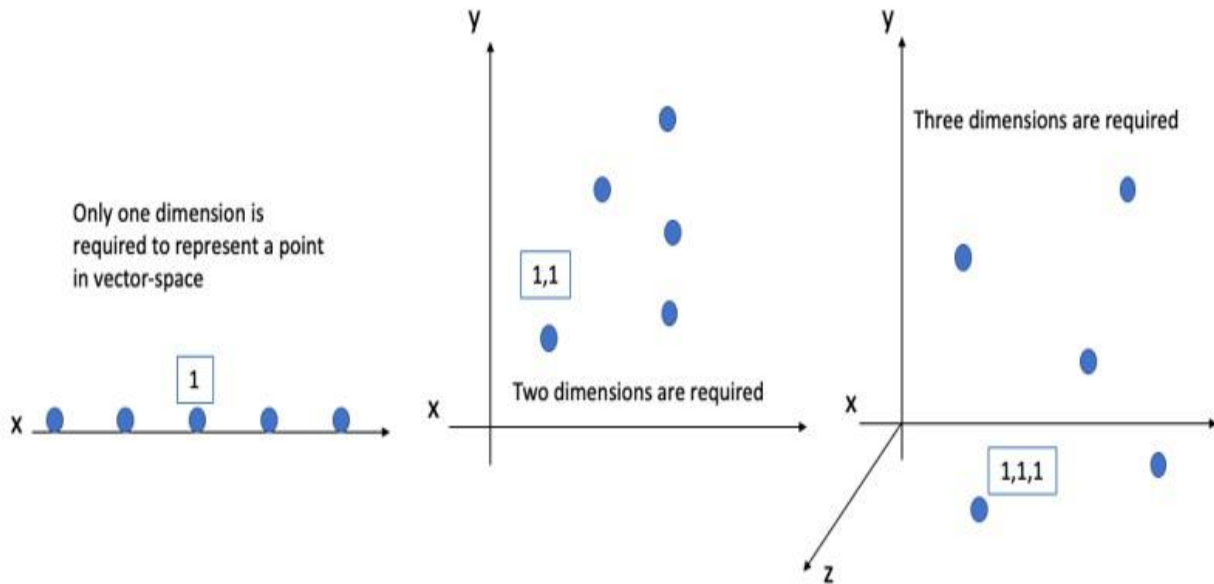
Now imagine if we have in total 20 data points to be analyzed. If we have only one attribute, we can represent it as x_1, x_2, x_3, x_4, x_5 …. x_{20} and hence a 20-dimension space will be sufficient to represent these points. in the second example where we require two dimensions, we will require (x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4), (x_5,y_5)….. (x_{20},y_{20}) or in other words 20*20 = 400 dimension space. For a three-dimensional one, we will represent a point as (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3), (x_4,y_4,z_4), (x_5,y_5,z_5)….. (x_{20},y_{20},z_{20}) and we will need 20*20*20 = 800 dimension space. And this process will continue.

Hence, it is quite easy for us to conclude that with increase in the number of dimensions, the amount of the space required to represent increases by leaps and bounds. It is referred to as the *Curse of Dimensionality*. The term was introduced by Richard E. Bellman and is used to refer to the problem of having too many variables in a dataset – some of which are significant while a lot may be of less importance.

There is another well-known theory known as *Hughes' Phenomenon* shown in (Figure 3-2). Generally, in data science and machine learning, we wish to have as much variables as possible to train our model. It is observed that the performance of the supervised learning classifier algorithm will increase to a certain limit and will peak with the most optimal number of variables. But, using the same amount of training data and with increased number of

dimensions, there is a decrease in the performance of a supervised classification algorithm. In other words, it is not advisable to have the variables in a dataset if they are not contributing to the accuracy of the solution. And we should remove such variables from the dataset.

Figure 3-2 Hughes phenomenon shows that the performance of a machine learning model will improve initially with increase in the number of dimensions. But with a further increase, it leads to a decrease in the model's performance.
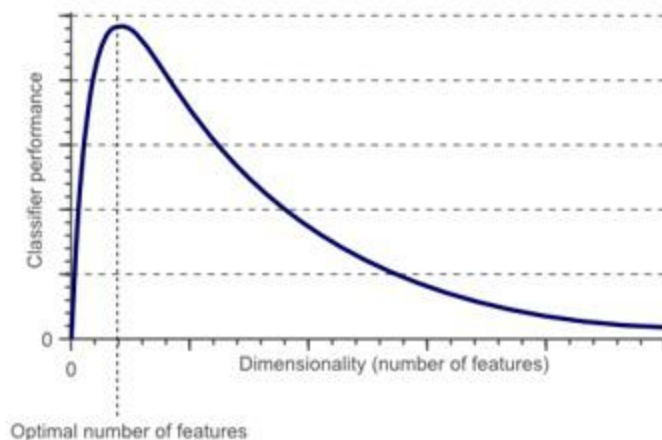
Increase in number of dimensions has the following impacts on the machine learning model:

1. As the model is dealing with an increased number of variables, the mathematical complexity increases. For example, in the case of the k-means clustering method we discussed in the last chapter – when we have a greater number of variables, the distance calculation between respective points will become complex. And hence, the overall model becomes more complex.
2. The dataset generated in a larger-dimensional space can be much sparser as compared to a smaller number of variables. The dataset will be sparser as some of the variables will have missing values, NULLs etc. The space is hence much emptier, the dataset is less dense, and a smaller number of variables have values associated with them.
3. With an increased complexity in the model, the processing time required increases. The system feels the pressure to deal with so many dimensions.
4. And the overall solution becomes more complex to comprehend and

execute. Recall from Chapter 1 where we have discussed supervised learning algorithms. Due to the high number of dimensions, we might face the problem of overfitting in supervised learning models.

When a supervised learning model has a good accuracy on training data but lesser accuracy on unseen data, it is referred as Overfitting. Overfitting is a nuisance as the very aim of machine learning models is to work well on unseen datasets and overfitting defeats this purpose.

Let us relate it to a real-world example. Consider an insurance company offering different types of insurance policies like life insurance, vehicle insurance, health insurance, home insurance etc. The company wishes to leverage data science and execute clustering use cases to increase the customer base and the total number of policies sold. They have customer details like age, gender, profession, policy amount, historical transactions, number of policies held, annual income, type of policy, number of historical defaults etc. At the same time, let us assume that variables like whether the customer is left-handed or right-handed, wears black or brown shoes, the shampoo brand used, colour of hair, favourite restaurant are also captured. If we include all the variables in the dataset, the total number of variables in the resultant dataset will be quite high. The distance calculation will be more complex for a k-means clustering algorithm, the processing time will increase and the overall solution will be quite a complex one.

It is also imperative to note that *not* all the dimensions or variables are significant. Hence, it is vital to filter the important ones from all the variables we have. Remember, nature always prefers simpler solutions! In the case discussed above, it is highly likely that variables like colour of the hair and favourite restaurant etc. will not impact the outputs. So, it is in our best interest to reduce the number of dimensions to ease the complexity and reduce the computation time. At the same time, it is also vital to note that dimensionality reduction is not always desired. It depends on the type of dataset and the business problem we wish to resolve. We will explore it more when we work on the case study in subsequent sections of the chapter.

🎯 POP QUIZ – answer these questions to check your understanding.. Answers at the end of the book

1. Curse of dimensionality refers to having a big size of data. TRUE or FALSE.

2. Having a high number of variables will always increase the accuracy of a solution. TRUE or FALSE.

3. How a large number of variables in a dataset impact the model?

We have established that having a lot of dimensions is a challenge for us. We are now examining the various methods to reduce the number of dimensions.

3.3 Dimension reduction methods

We studied the disadvantages of having a really high dimensional data in the last section. A lesser number of dimensions might result into a simpler structure which will be computationally efficient. At the same time, we should be careful with reducing the number of variables. The output of dimension reduction method should be complete enough to represent the original data and should not lead to any information loss. In other words, if originally, we had for example 500 variables and we reduced it to 120 significant ones, still these 120 *should* be robust enough to capture *almost* all the information. Let us understand using a simple example.

Consider this, we wish to predict the amount of rainfall a city will receive in the next month. The rainfall prediction for that city might be dependent on temperature over a period of time, wind speed measurements, pressure, distance from the sea, elevation above the sea level etc. These variables make sense if we wish to predict the rainfall. At the same time, variables like number of cinema halls in the city, whether the city is a capital of the country or number of red cars in the city might not impact the prediction of rainfall. In such a case, if we do not use the number of cinema halls in the city to predict the amount of rainfall, it will not reduce the capability of the system. The solution in all probability, will be still able to perform quite well. And hence, in such a case no information will be lost by dropping such a variable

and surely, we can drop it from the dataset. It is a very simple example to highlight the need to reduce the number of variables.

The dimensions or the number of variables can be reduced by a combination of manual and algorithm-based methods. But before studying them in detail, there are a few mathematical terms and components which we should be aware of before proceeding ahead, which we are discussing now.

3.3.1 Mathematical foundation

There are quite a few mathematical terms which one must grab to develop a thorough understanding of dimensionality reduction methods. These mathematical terms are intuitive enough and you might have covered them in your earlier mathematical courses.

We are trying to reduce the number of dimensions of a dataset. A dataset is nothing but a matrix of values – hence a lot of the concepts are related to matrix manipulation methods, geometrical representation of them and performing transformations on such matrices. The mathematical concepts are discussed in the Appendix Mathematical Foundation of the book. You would also require understanding of eigen values and eigen vectors. These concepts will be reused throughout the book and hence they have been put in the Appendix for quick reference. You are advised to go through them before proceeding.We will now explore a few manual methods for dimensionality reduction methods and then proceed to algorithm-based ones.

3.4 Manual methods of dimensionality reduction

To tackle curse of dimensionality, we wish to reduce the number of variables in a dataset. The reduction can be done by removing the variables from the dataset. Or a very simple solution for dimensionality reduction can be combining the variables which can be grouped logically or can be represented using a common mathematical operation.

For example, as shown in (Table 3-5) below, the data can be of a retail store where different customers have generated different transactions. We will get the sales, number of invoices and the number of items bought for each

customer over a period of time. In the table below, customer 1 has generated two invoices, bought 5 items in total and generated a total sale of 100.

If we wish to reduce the number of variables, we might combine three variables as two variables. Here, we have introduced variables ATV (average transaction value) and ABS (average basket size) wherein ATV = Sales/Invoices and ABS = NumberOfItems/Invoices.

So, in the second table for Customer 1, we have ATV as 50 and ABS as 2.5. Hence, the number of variables has been reduced from three to two.

Table 3-5 In the first table, we have the sales, invoices and number of items as the variables. In the second table, they have been combined to create new variables.

CustomerID	Sales	Invoices	NumberOfItems	CustomerID	ATV	ABS
1	100	2	5	1	50	2.5
2	200	2	4	2	100	2
3	300	10	12	3	30	1.2
4	400	2	10	4	200	5
5	500	5	12	5	100	2.4

And this process can continue to reduce the number of variables. Similarly for a telecom subscriber, we will have the minutes of mobile calls made during 30 days in a month. We can add them to create a single variable – *minutes used in a month*. The above examples are very basic ones to start with. Using manual process, we can employ two other commonly used methods – manual selection and using correlation coefficient.

Manual feature selection

Continuing from the rainfall prediction example we discussed in the last section – a data scientist might be able to drop a few variables. This will be based on the deep understanding of the business problem at hand and the corresponding dataset being used. However, it is an underlying assumption that the dataset is quite comprehensible for the data scientist and they understand the business domain well. Most of the time, the business stakeholders will be able to guide on such methods. It is also necessary that

the variables are unique and not much of a dependency exists.

As shown in (Table 3-6) below, we can remove a few of the variables which might not be useful for predicting rainfall.

Table 3-6 In the first table, we have all the variables present in the dataset. Using business logic, some of the variables which might be not much use have been discarded in the second table. But this is to be done with due caution. And the best way is to get guidance from the business stakeholders.

Temperature	Pressure	Elevation	Is_capital	Number of cars	Distance from the sea	Number of malls	Number of parks
50	1.1	200	Y	1000	100	5	4
51	1.2	200	N	1200	120	4	6
52	1.1	200	Y	1100	150	5	8
53	1.2	200	N	2000	200	2	4
54	1.2	200	Y	2100	120	6	2

Temperature	Pressure	Elevation	Distance from the sea
50	1.1	200	100
51	1.2	200	120
52	1.1	200	150
53	1.2	200	200
54	1.2	200	120

Sometimes, feature selection methods are also referred to as *wrapper methods*. Here, a machine learning model is wrapped or fitted with a subset of variables. In each iteration, we will get a different set of results. The set which generates the best results is selected for the final model.

The next methods are based on the correlation existing between various attributes.

Correlation coefficient

Correlation between two variables simply mean that have a mutual relationship with each other. The change in the value of one variable will impact the value of other. The variables which are highly correlated with each other are supplying similar information and hence one of them can be dropped.

Correlation is described in detail in the Appendix Mathematical Foundation of the book.

For example, for a retail store the number of invoices generated in a day will be highly correlated with the amount of sales generated and hence, one of them can be dropped. Another example can be – students who study for a higher number of hours will have better grades than the ones who study less

(mostly!).

But we should be careful in dropping the variables and should not trust correlation alone. The business context of a variable should be thoroughly understood before taking any decision.

It is a good idea to discuss with the business stakeholders before dropping any variables from the study.

Correlation based methods are sometimes called *filter methods*. Using correlation coefficients, we can filter and choose the variables which are most significant.

🎯 POP QUIZ – answer these questions to check your understanding.. Answers at the end of the book

1. We can drop a variable simply if we feel is not required. TRUE or FALSE.

2. If two variables are correlated, ALWAYS drop one of them. TRUE or FALSE.

Manual methods are easier solutions and can be executed quite efficiently. The dataset size is reduced and we can proceed ahead with the analysis. But manual methods are sometimes subjective and depend a lot on the business problem at hand. Many times, it is also not possible to employ manual methods for dimension reduction. In such situations, we have algorithm-based methods which we are studying in the next section.

3.4.1 Algorithm based methods for reducing dimensions

We examined manual methods in the last section. Continuing from there, we will examine algorithm-based methods in this section. The algorithm-based techniques are based on a more mathematical base and hence prove to be more scientific methods. In real-world business problems, we use a combination of both manual and algorithm-based techniques. Manual

methods are straightforward to execute as compared to algorithm-based techniques. Also, we cannot comment on the comparison of both techniques, as they are based on different foundations. But at the same time, it is imperative that you put due diligence in the implementation of algorithm-based techniques.

The major techniques used in dimensionality reductions are listed below. We will be exploring most of them in this book.

1. Principal Component Analysis (PCA)
2. Singular Value Decomposition (SVD)
3. Linear Discriminant Analysis (LDA)
4. Generalized Discriminant Analysis (GDA)
5. Non-negative matrix factorization (NMF)
6. Multi-dimension scaling (MDS)
7. Locally Linear Embeddings (LLE)
8. IsoMaps
9. Autoencoders
10. t-SNE (T-distributed Stochastic Neighbor Embedding)

These techniques are utilized for the common end goal – transform the data from a high dimensional space to a low dimensional one. Some of the data transformations are linear in nature, while some are non-linear.

We are going to discuss Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) in detail in this chapter. In the later chapters of the book, other major techniques are being explored. The reason to choose PCA and SVD is owing to their popularity. They might not offer the best solution in all the situations, but are quite a robust methods and heavily used. Perhaps, PCA is the most quoted dimensionality reduction method which we are exploring in the next section.

3.5 Principal Component Analysis (PCA)

Consider this. You are working on a dataset which has 250 variables. It is almost impossible to visualize such a high-dimensional space. Some of the 250 variables might be correlated with each other, some of them might not be

and there is a need to reduce the number of variables without losing much information. Principal Component Analysis or PCA allows us to mathematically select the most important features and leave the rest. PCA does reduce the number of dimensions but also preserves the most important relationships between the variables and the important structures in the dataset. Hence, the number of variables is reduced but the important information in the dataset is kept safe.

PCA is a projection of high-dimensional data in lower dimensions. In simpler terms, we are reducing a n dimensional space into an m dimensional one where n > m while maintaining the nature and the essence of the original dataset. In the process, the old variables are reduced to newer ones, while maintaining the crux of the original dataset The new variables thus created are called *Principal Components*. The principal components are a linear combination of the raw variables. As a result of this transformation, the first principal component captures the maximum randomness or the highest variance in the dataset. The second principal component created is orthogonal to the first component.

If two straight lines are orthogonal to each other, it means they are at an angle of 90^0 to each other,

And the process continues to the third component and so on. Orthogonality allows us to maintain that there is no correlation between subsequent principal components.

PCA utilizes linear transformation of the dataset and such methods are sometimes referred to as feature projections. The resultant dataset or the projection is used for further analysis.

Let us understand better by means of an example. In the (Table 3-7) shown below, we have represented the total perceived value from a home using some variables. The variables are area (sq m), number of bedrooms, number of balconies, distance from the airport, distance from the train station and so on – we have 100+ variables.

Table 3-7 The variables based on which a price of a house can be estimated

Area (sq m)	Number of bedrooms	Number of balconys	Distance from airport	No of schools	...and so on
100	2	2	20	2	
200	3	2	21	4	
250	4	4	16	2	
400	4	3	15	5	
450	5	4	25	4	

We can combine some of the variables mathematically and logically. PCA will create a new variable which is a linear combination of some of the variables as shown in the example below. It will get the best *linear* combination of original variables so that the new_variable is able to capture the maximum variance of the dataset. The Equation 3-7 is only an example shown for illustration purpose wherein we are showing a new_variable created by a combination of other variables.

new_variable = a*area – b*bedrooms + c*distance – d*schools

(Equation 3-7)

Now let's understand the concept visually. In a vector-space diagram, we can represent the dataset as shown in (Figure 3-4) below. The first figure represents the raw data where we can visualize the variables in a x-y diagram. As discussed above, we wish to create a linear combination of variables. Or in other words, we wish to create a mathematical equation which will be able to explain the relationship between x and y.

The output of such a process will be a straight line as shown in the second figure in (Figure 3-4). This straight line is sometimes referred to as the *Line of Best fit.* Using this line of best fit, we can predict a value of y for a given value of x. These predictions are nothing but the projections of data points on the straight line.

The difference between the actual value and the projections is the error as shown in the third figure in (Figure 3-4) below). The total sum of these errors is called total projection error.

Figure 3-4 (i) The dataset can be represented in a vector-space diagram (ii) The straight line can be called as the line of best fit having the projections of all the data points on it. (iii) The difference between the actual value and the projections are the error terms.
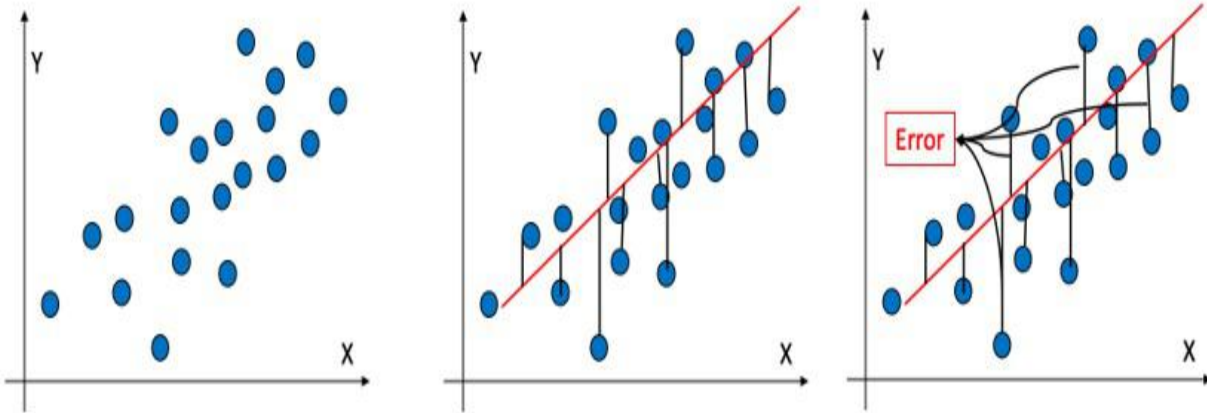
There can be multiple options for this straight line as shown in (Figure 3-5) below. These different straight lines will have different errors and different values of variances captured.

Figure 3-5 The data set can be captured by a number of lines, but not all the straight lines will be able to capture the maximum variance. The equation which gives the minimum error will be the chosen one.
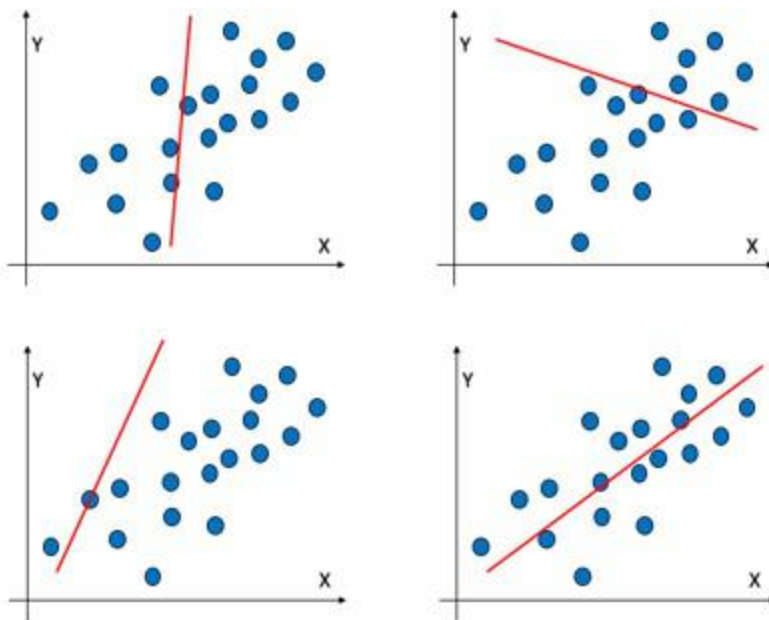

The straight line which is able to capture the maximum variance will be chosen one. Or in other words, it is giving the minimum error. It will be the *first principal component* and the direction of maximum spread will be the *principal axis*.

The second principal component will be derived in a similar fashion. Since we know the first principal axis, we can subtract the variance along this principal axis from the total variance to get the residual variance. Or in other words, using the first principal component we would capture some variance in the dataset. But there will be a portion of total variance in the dataset which is still unexplained by the first principal component. The portion of total variance unexplained is the residual variance. Using the second principal component, we wish to capture as much variance as we can.

Using the same process to capture the direction of maximum variance, we will get the second principal component. The second principal component can be at a number of angles with respect to the first one as shown in (Figure 3-6). It is mathematically proven that if the second principal component is orthogonal to the first principal component, it allows us to capture the maximum variance using the two principal components. In (Figure 3-6), we can observe that the two principal components are at an angle of 90^0 with each other.

Figure 3-6 (i) The first figure on the left is the first principal component. (ii) The second principal component can be at different angles with respect to the first principal component. We have to find the second principal which allows to capture the maximum variance (iii) To capture the maximum variance, the second principal component should be orthogonal to the first one and hence the combined variance captured in maximized.
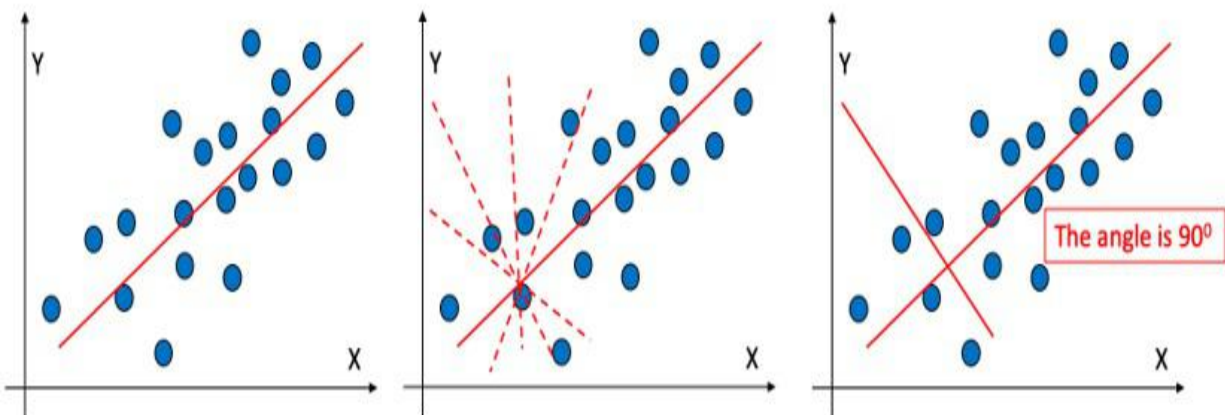
And the process continues for the third, fourth principal component and so on. Once all the principal components are derived, the dataset is projected onto these axes. The columns in this transformed dataset are the *principal components*. The principal components created will be lesser than the number

of original variables and capturing maximum information present in the dataset.

Before we examine the process of PCA in depth, let's study its important characteristics:

1. PCA aims to reduce the number of dimensions in the resultant dataset.
2. PCA produces principal components which aim to reduce the noise in the dataset by maximizing the feature variance.
3. At the same time, the principal components reduce the redundancy in the dataset. It is achieved by minimizing the covariance between the pairs of features.
4. The original variables no longer exist in the newly created dataset. Instead, new variables are created using these variables.
5. It is not necessary that the principal components will map one-to-one with all the variables present in the dataset. They are a new combination of the existing variables. And hence, they can be a combination of several different variables in one principal component (as shown in Equation 3-7).
6. The new features created from the dataset do not share the same column names.
7. The original variables might be correlated with each other, but the newly created variables are unrelated with each other.
8. The number of newly created variables is lesser than the original number of variables. After all, that is the whole purpose of dimensionality reduction.
9. If PCA has been used for reducing the number of variables in a training dataset, the testing/validation datasets have to be reduced too using PCA.
10. PCA is not synonyms to dimensionality reduction. It can be put into use for a number of other usages too. It is generally a PCA only for dimensionality reduction will be a misnomer for sure.

We will now examine the approach used while implementing PCA and then we will develop a Python solution using PCA. Though we need not apply all the steps while we develop the codes, as the heavy lifting has already been done by the packages and libraries. The steps given below are hence taken

care of by the packages, but still it is imperative that you understand these steps to appreciate the full understanding of PCA.

The steps followed in PCA are:

1. In PCA, we start with **normalizing our dataset** as a first step. It ensures that all our variables have a common representation and become comparable. We have methods to perform the normalization in Python, which we will study when we develop the code. To explore more on normalizing the dataset, you can refer to the Appendix Mathematical Foundation.
2. **Get the covariance in** the normalized dataset. It allows us to study the relationship between the variables. We generally create a covariance matrix as shown in the Python example in the next section.
3. We can then calculate the eigenvectors and eigenvalues of the covariance matrix.
4. We then sort the eigenvalues in the decreasing order of eigenvalues. The eigenvectors corresponding to the maximum value of eigenvalues are chosen. The components hence chosen will be able to capture the maximum variance in the dataset. There are other methods to shortlist the principal components which we will explore while we develop the Python code.

🎯 POP QUIZ – answer these questions to check your understanding.. Answers at the end of the book

1. PCA will result in the same number of variables in the dataset. TRUE or FALSE.

2. PCA will be able to capture 100% information in the dataset. TRUE or FALSE.

3. What is the logic of selecting principal components in PCA?

So, in essence, principal components are the linear combinations of the original variables. The weight in this linear combination is the eigenvector

satisfying the error criteria of the least square method. We are studying Eigenvalue decomposition now and SVD is covered in the next section.

3.5.1 Eigenvalue Decomposition

We studied PCA in the last section where we said that principal components are the linear combination of the original variables. We will explore on the eigenvalue decomposition for PCA now.

In the context of PCA, the eigenvector will represent the direction of the vector and the eigenvalue will be the variance that is captured along that eigenvector. It can be shown by means of the (Figure 3-7) below, where we are breaking the original nxn matrix into components.

Figure 3-7 Using Eigenvalue decomposition, the original matrix can be broken into eigenvector matrix, eigenvalue matrix and an inverse of eigenvector matrix. We implement PCA using this methodology.

Original n x n matrix   Eigenvector matrix   Eigenvalue matrix   Inverse of the eigenvector matrix
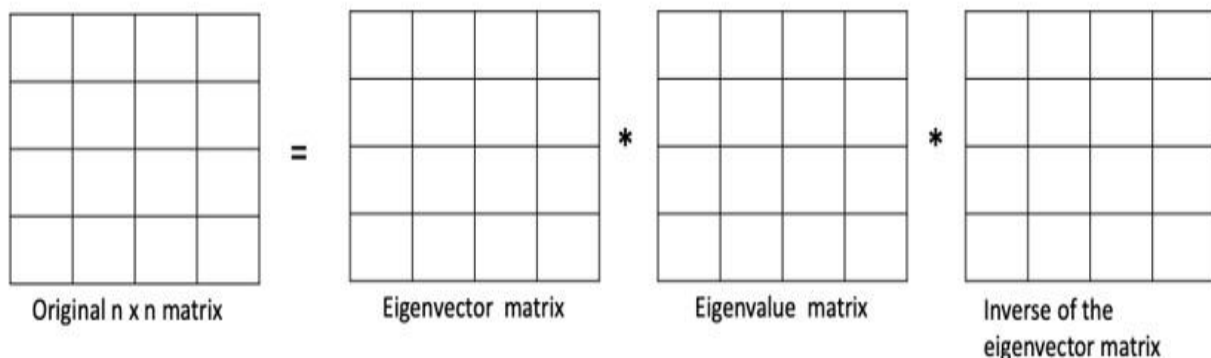
Mathematically, we can show the relation by (Equation 3-8)

$$A*v = \lambda*v$$

(Equation 3-8)

where A is a square matrix, v is the eigenvector and λ is the eigenvalue. Here, it is important to note that Eigenvector matrix is the orthonormal matrix and its columns are eigenvectors. Eigenvalue matrix is the diagonal matrix and its eigenvalues are the diagonal elements. The last component is the inverse of

the eigenvector matrix. Once we have the eigenvalues and the eigenvectors, we can choose the significant eigenvectors for getting the principal components.

We are presenting PCA and SVD as two separate methods in this book. Both of the methods are used to reduce high-dimensional data into fewer ones, and in the process retain the maximum information in the dataset. The difference between the two is – SVD exists for any sort of matrix (rectangular or square), whereas the eigen decomposition is possible only for square matrices. You will understand it better once we have covered SVD in the later part of this chapter.

We will now create a Python solution using eigenvalue decomposition.

3.5.2 Python solution using PCA

We have studied the concepts of PCA and the process using eigenvalue decomposition. It is time for us to dive into Python and develop a PCA solution on a dataset. We will show you how to create eigenvectors and eigen values on the dataset. To implement the PCA algorithms, we will use the `sklearn` library. Libraries and packages provide a faster solution for implementing algorithms.

We are using the Iris dataset for this problem. It is one of the most popular datasets used for machine learning problems. The dataset contains data of three iris species with 50 samples each and having properties of each flower – like petal length, sepal length etc. The objective of the problem is to predict the species using the properties of the flower. The independent variables hence are the flower properties whereas the variable "Species" is the target variable. The dataset and the code are checked-in at the Github repository. Here, we are using the inbuilt PCA functions which reduce the effort required to implement PCA.

Step1: Load all the necessary libraries first. We are going to use numpy, pandas, seaborn, matplotlib and sklearn. Note that we have imported PCA from sklearn.

These are the most standard libraries. You will find that almost all the

machine learning solutions would import these libraries in the solution notebook.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

Step 2: Load the dataset now. It is a .csv file.

```
iris_df = pd.read_csv('iris.csv')
```

Step 3: We will now perform a basic check on the dataset – looking at the first five rows, shape of the data, spread of the variables etc. We are not performing an extensive EDA here as the steps are covered in Chapter 2. The dataset has 150 rows and 6 columns.

```
iris_df.head()
```

iris_df.head()

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
iris_df.describe()
```

iris_df.describe()

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
iris_df.shape
```

Step 4: Here we have to break the dataset into independent variables and target variable. X_variables here represent the independent variables which are in the first 4 columns of the dataset while y_variable is the target variable which is Species in this case and is the final column in the dataset. Recall, we wish to predict the Species of a flower using the other properties. Hence we have separated the target variable Species and other independent variables.

```
X_variables = iris_df.iloc[:,1:5]
X_variables
y_variable = iris_df.iloc[:,5]
```

Step 5: We are now normalizing our dataset. The inbuilt method of StandardScalar() does the job for us quite easily.

StandardScalar method normalizes the dataset for us. It subtracts the mean from the variable and divided by the standard deviation. For more details on normalization, refer to the Appendix Mathematical Foundation.

We invoke the method and then use it on our dataset to get the transformed dataset. Since, we are working on independent variables, we are using X_variables here. First we invoke the StandardScalar() method. And then we use fit_transform method. The fit_transform method first fits the transformers to X and Y and then returns a transformed version of X.

```
sc = StandardScaler()
transformed_df = sc.fit_transform(X_variables)
```

Step 6: We will now calculate the covariance matrix. And print it, the output is shown below. Getting the covariance matrix is straightforward using numpy.

```
covariance_matrix = np.cov(transformed_df.T)
covariance_matrix
```

```
array([[ 1.00671141, -0.11010327,  0.87760486,  0.82344326],
       [-0.11010327,  1.00671141, -0.42333835, -0.358937  ],
       [ 0.87760486, -0.42333835,  1.00671141,  0.96921855],
       [ 0.82344326, -0.358937  ,  0.96921855,  1.00671141]])
```

Step 7: Now in this step the eigenvalues are being calculated. Inside numpy library we have the inbuilt functionality to calculate the eigenvalues. We will then sort the eigenvalues in descending order. To shortlist the principal components, we can choose eigenvalues greater than 1. This criterion is called *Kaiser criteria.* We are exploring other methods too.

Eigenvalue represents how much a component is good as a summary of the data. If the eigenvalue is 1, it means that the component contains same amount of information as a single variable. And hence we choose the eigenvalue which is greater than 1.

In this code, first we are getting the `eigen_values` and `eigen_vectors`. And then we are arranging then in descending order.

```
eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)
eigen_pairs = [(np.abs(eigen_values[i]), eigen_vectors[:,i]) for
print('Eigenvalues arranged in descending order:')
for i in eigen_pairs:
    print(i[0])
```

```
Eigenvalues arranged in descending order:
2.9303537755893165
0.9274036215173417
0.1483422264816399
0.02074601399559571
```

Step 8: We will now invoke the PCA method from the sklearn library. The method is used to fit the data here. To be noted is, we have not yet determined the number of principal components we wish to use in this problem yet.

```
pca = PCA()
pca = pca.fit(transformed_df)
```

Step 9: The principal components are now set. Let's have a look at the variance explained by them. We can observe that the first component captures 72.77% variation, second captures 23.03% variation and so on.
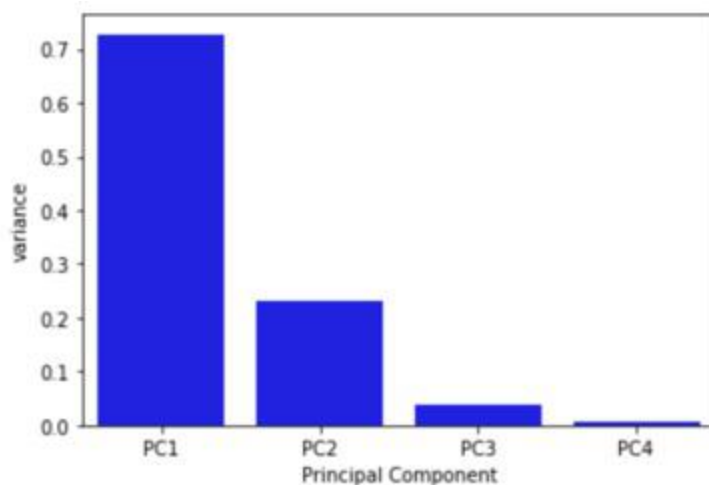
```
explained_variance = pca.explained_variance_ratio_
```

```
explained_variance
```

```
array([0.72770452, 0.23030523, 0.03683832, 0.00515193])
```

Step 10: We are now plotting the components in a bar plot for better visualization.

```
dataframe = pd.DataFrame({'var':pca.explained_variance_ratio_,
              'PC':['PC1','PC2','PC3','PC4']})
sns.barplot(x='PC',y="var",
          data=dataframe, color="b");
```


Step 11: We are drawing a scree-plot to visualize the cumulative variance being explained by the principal components.

```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```
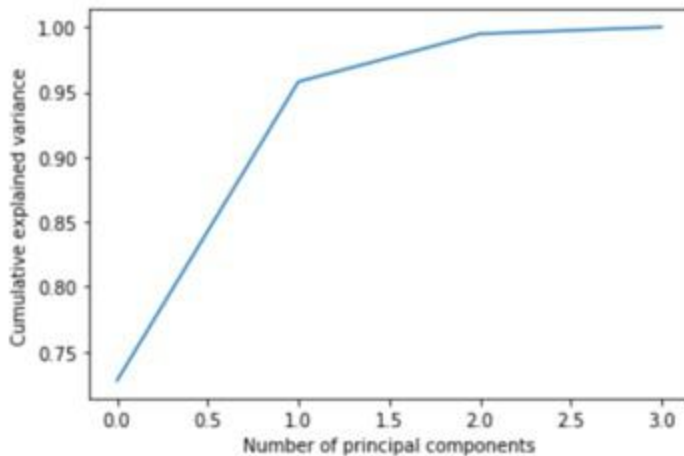
Step 12: In this case study, if we choose the top two principal components as the final solutions as these two capture 95.08% of the total variance in the dataset.

```
pca_2 = PCA(n_components =2 )
pca_2 = pca_2.fit(transformed_df)
pca_2d = pca_2.transform(X_variables)
```

Step 13: We will now plot the dataset with respect to two principal components. For that, it is a requirement that Species are tied-back to the actual values of the Species variable which are Iris-setosa, Iris-versicolor and Iris-virginica. Here, 0 is mapped to Iris-setosa, 1 is Iris-versicolor and 2 is Iris-virginica. In the code below, first the Species variable gets its values replaced by using the mapping discussed above.

```
iris_df['Species'] = iris_df['Species'].replace({'Iris-setosa':0,
```

Step 14: We will now plot the results with respect to two principal components. The plot is showing the dataset reduced to two principal components we have just created. These principal components are able to capture 95.08% variance of the dataset. The first principal component represents the x-axis in the plot while the second principal component represents the y-axis in the plot. The colour represents the various classes of Species.

```
plt.figure(figsize=(8,6))
plt.scatter(pca_2d[:,0], pca_2d[:,1],c=iris_df['Species'])
plt.show()
```
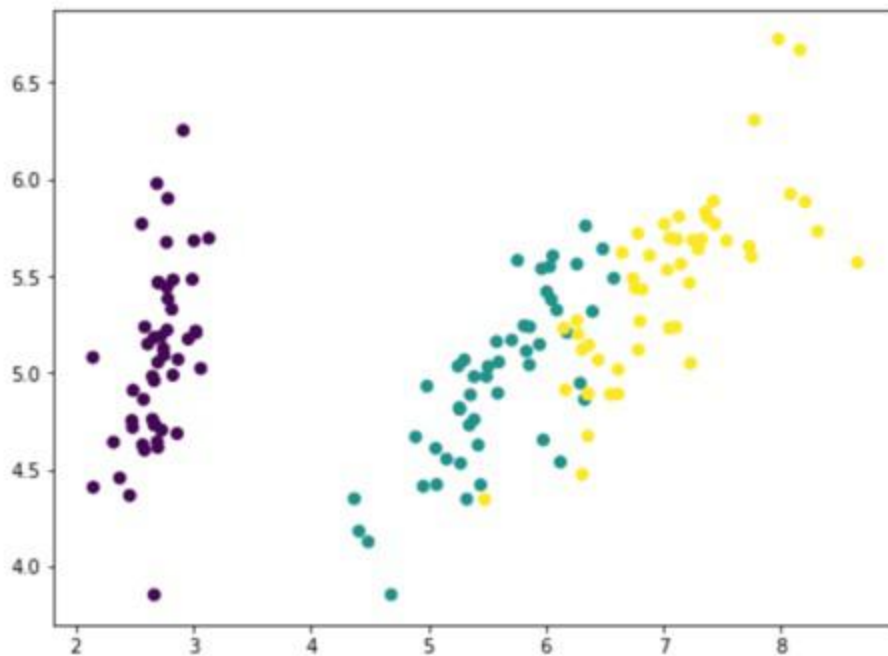
The above solution has reduced the number of components from four to 2 and still is able to retain most of the information. Here, we have examined three approaches to select the principal components – based on Kaiser criteria, the variance captured and scree plot.

Let us quickly analyse what we have achieved using PCA. The (Figure 3-8) shows two representations of the same dataset. The one on the left is the original dataset of X_variables. It has four variables and 150 rows. The right is the output of PCA. It has 150 rows but only two variables. Recall, we have reduced the number of dimensions from four to two. So the number of observations have remained as 150 only while the number of variables have reduced from four to two.

Figure 3-8 The figure on the left shows the original dataset which has 150 rows and 4 variables. After the implementation of PCA, the number of variables has been reduced to two. The number of rows remain the same as 150 which is shown by the length of pca_2d.

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
pca_2d[0:5]

array([[2.66923088, 5.18088722],
       [2.69643401, 4.6436453 ],
       [2.4811633 , 4.75218345],
       [2.57151243, 4.62661492],
       [2.59065822, 5.23621104]])
```

```
len(pca_2d)

150
```

Once we have reduced the number of components, we can continue to implement a supervised learning or an unsupervised learning solution. We can implement the above solution for any of the other real-world problems where we aim to reduce the number of dimensions. You will be exploring more on it in the case study section.

With this we have covered PCA. We will now explore Singular Value Decomposition (SVD) in the next section.

3.6 Singular Value Decomposition (SVD)

In the last section we studied PCA. PCA transforms the data linearly and generate principal components which are not correlated with each other. But the process followed of eigenvalue decomposition can only be applied to *square matrices*. Whereas SVD can be implemented to any m x n matrix. We will study this in more detail now.

Let us consider we have a matrix A. The shape of A is m x n or it contains m rows and n columns. The transpose of A can be represented as A^T.

We can create two other matrices using A and A^T as *A A*T and *ATA*. These resultant matrices *A A*T and *ATA* have some special properties which are listed below. The mathematical proof of the properties is beyond the scope of the book.

The properties of AA^T and A^TA are:

1. They are symmetric and square matrices.
2. Their eigenvalues are either positive or zero.
3. Both AA^T and A^TA are having the same eigenvalue.
4. Both AA^T and A^TA are having same rank as the original matrix A.

The eigenvectors of AA^T and A^TA are referred to as **singular vectors of A**. The square root of their eigenvalues is called **singular values**.

Since both of these matrices (AA^T and A^TA) are symmetrical, their eigenvectors can be orthonormal to each other. In other words, being symmetrical - the eigenvectors can be perpendicular to each other and can be of unit length.

Now, with these mathematical understanding we can define SVD. As per Singular Value Decomposition method, it is possible to factorize any matrix A as

A = U * S * VT
(Equation 3-9)

Here, A is the original matrix,

U and V are the orthogonal matrices with orthonormal eigenvectors taken from AA^T and A^TA respectively and

S is the diagonal matrix with r elements equal to the singular values.

In simple terms, SVD can be said as an enhancement of the PCA methodology using eigenvalue decomposition.

Singular values are better and numerically more robust than eigenvalues decomposition.

PCA was defined as the linear transformation of input variables using principal components. All those concepts of linear transformation, choosing the best components etc. remain the same. The major process steps remain

similar, except in SVD we are using a slightly different approach wherein the eigenvalue decomposition has been replaced with using singular vectors and singular values. It is often advisable to use SVD when we have a sparse dataset, in the case of denser dataset PCA can be utilized.

🎯 POP QUIZ – answer these questions to check your understanding.. Answers at the end of the book

1. SVD works on eigenvalue decomposition technique. TRUE or FALSE.

2. PCA is a much more robust methodology than SVD. TRUE or FALSE.

3. What are singular values and singular vectors in SVD?

We will now create Python solution using SVD in the next section.

3.6.1 Python solution using SVD

In this case study, we are using mushrooms dataset. This dataset contains descriptions of 23 species of grilled mushrooms. There are two classes – either the mushroom is "e" which means it is edible else the mushroom is "p" meaning it is poisonous.

Step 1: Import the libraries. We are importing

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

Step 2: Import the dataset and check for shape, head etc.

```
mushrooms_df = pd.read_csv('mushrooms.csv')
mushrooms_df.shape
mushrooms_df.head()
```

class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	h
p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	s	
e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	n	
e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	n	
p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	s	
e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	a	

Step 3: As we can observe, the values are categorical in nature in the dataset. They have to be first encoded into numeric values. This is not the only approach we deal with categorical variables. There are other techniques too which we will explore throughout the book. We will dive deep into the techniques in the last chapter of the book.

First, we are invoking the LabelEncoder and then apply it to all the columns in the dataset. The LabelEncoder converts the categorical variables into numeric ones using one-hot encoding method.

```
encoder = LabelEncoder()
for col in mushrooms_df.columns:
    mushrooms_df[col] = encoder.fit_transform(mushrooms_df[col])
```

Step 4: Have a re-look at the dataset. All the categorical values have been converted to numeric ones.

```
mushrooms_df.head()
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population
0	1	5	2	4	1	6	1	0	1	4	...	2	7	7	0	2	1	4	2	3
1	0	5	2	9	1	0	1	0	0	4	...	2	7	7	0	2	1	4	3	2
2	0	0	2	8	1	3	1	0	0	5	...	2	7	7	0	2	1	4	3	2
3	1	5	3	8	1	6	1	0	1	5	...	2	7	7	0	2	1	4	2	3
4	0	5	2	3	0	5	1	1	0	4	...	2	7	7	0	2	1	0	3	0

Step 5: The next two steps are same as the last case study wherein we break the dataset into X_variables and y_label. And then the dataset is normalized.

```
X_variables = mushrooms_df.iloc[:,1:23]
y_label = mushrooms_df.iloc[:, 0]
scaler = StandardScaler()
X_features = scaler.fit_transform(X_variables)
```

Step 6: In this step, we implement the SVD. There is a method in numpy which implements SVD. The output is u, s and v, where u and v are the singular vectors and s is the singular value. If you wish you can analyse their respective shapes and dimensions.

```
u, s, v = np.linalg.svd(X_features, full_matrices=True)
```

Step 7: We know that singular values allow us to compute variance explained by each of the singular vectors. We will now analyze the % variance explained by each singular vector and plot it. The results are being shown till three places of decimal. And then we are plotting the results as a histogram plot. On the x-axis we have the singular vectors while on the y-axis we have the % of variance explained.

```
variance_explained = np.round(s**2/np.sum(s**2), decimals=3)
variance_explained
sns.barplot(x=list(range(1,len(variance_explained)+1)),
            y=variance_explained, color="blue")
plt.xlabel('SVs', fontsize=16)
plt.ylabel('Percent of the variance explained', fontsize=15)
```

Text(0, 0.5, 'Percent of the variance explained')
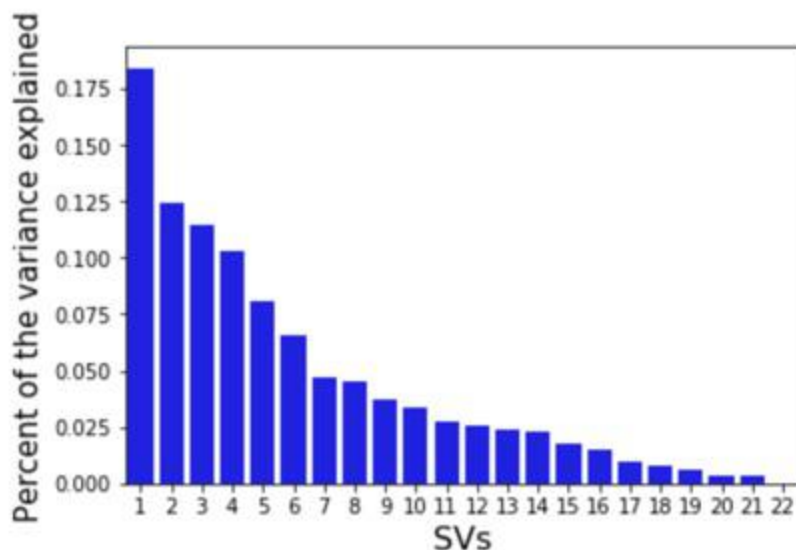
Step 7: We will now create a data frame. This new data frame svd_df contains the first two singular vectors and the metadata. We are then printing the first 5 rows using the head command.

```
col_labels= ['SV'+str(i) for i in range(1,3)]
svd_df = pd.DataFrame(u[:,0:2], index=mushrooms_df["class"].tolis
svd_df=svd_df.reset_index()
svd_df.rename(columns={'index':'Class'}, inplace=True)
svd_df.head()
```

	Class	SV1	SV2
0	1	0.003238	-0.006710
1	0	0.012864	0.001919
2	0	0.010474	-0.001863
3	1	0.004988	-0.005202
4	0	-0.003887	0.008522

Step 8: Similar to the last case study, we are replacing numeric values with actual class labels. 1 is edible while 0 is Poisonous.

```
svd_df['Class'] = svd_df['Class'].replace({1:'Edible', 0:'Poison'
```

Step 9: We are now plotting the variance explained by the two components. Here we have chosen only the first two components. You are advised to take optimum number of components using the methods described in the last section and plot the respective scatter plots. Here, on the x-axis we have shown the first singular vector SV1 and on the y-axis we have shown the second singular vector SV2.

```
color_dict = dict({'Edible':'Black',
                   'Poison': 'Red'})
sns.scatterplot(x="SV1", y="SV2", hue="Class",
               palette=color_dict,
               data=svd_df, s=105,
               alpha=0.5)
plt.xlabel('SV 1: {0}%'.format(variance_explained[0]*100), fontsi
plt.ylabel('SV 2: {0}%'.format(variance_explained[1]*100), fontsi
```
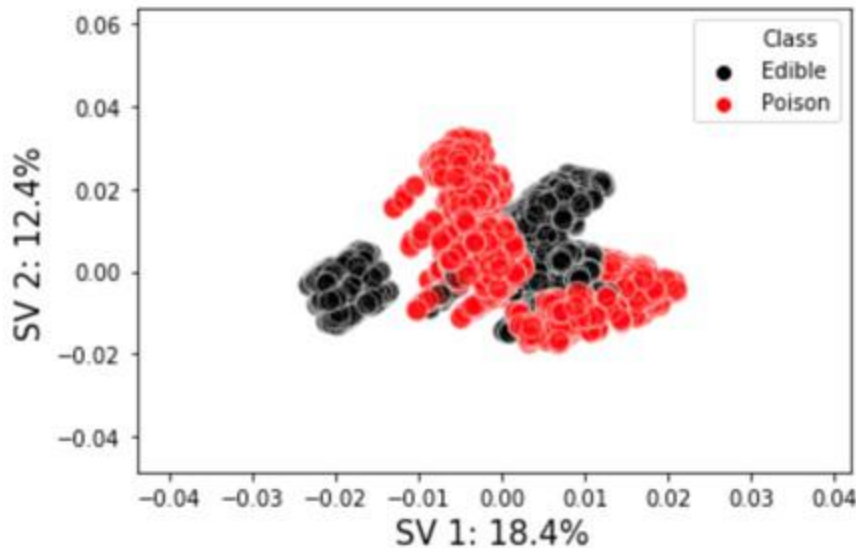
We can observe the distribution of the two classes with respect to the two components. The two classes – edible and poison are colour coded as black and red respectively. As we have noted above, we have chosen only two components to show the impact using a visualization plot. You are advised to choose the optimum number of components using the methods described in the last case study and then visualize the results using different singular vectors. This solution can be used to reduce dimensions in a real-world dataset.

This concludes our discussion on SVD. We will now observe the positives and challenges with PCA and SVD in the next section.

3.7 Pros and cons of dimensionality reduction

In the initial sections of the chapter, we discussed the drawbacks the of curse of dimensionality. In the last few sections, we discovered PCA and SVD and implemented using Python. In the current section, we will examine the advantages and challenges we have with these techniques.

The major advantages we get with implementing PCA or SVD are as:

1. **Reduced number of dimensions** lead to less complexity in the dataset. The correlated features are removed and are transformed. Treating

correlated variables manually is a tough task which is quite manual and frustrating. Techniques like PCA and SVD do that job for us quite easily. The number of correlated features is minimized, and overall dimensions are reduced.

2. **Visualization of the dataset** is better if the number of dimensions is lesser. It is very difficult to visualize and depict a very-high dimensional dataset.
3. **Accuracy** of the machine learning model is improved if the correlated variables are removed. These variables do not add anything to the performance of the model.
4. **The training time is reduced** as the dataset is less complex. Hence, lesser computation power is required, and lesser time is required.
5. **Overfitting** is a nuisance in supervised machine learning models. It is a condition where the model is behaving very well on the training dataset but not so well on the testing/validation dataset. It means that the model may not be able to perform well on the real-world unseen datasets. And it beats the entire purpose of building the machine learning model. PCA/SVD helps in tackling overfitting by reducing the number of variables.

At the same time, there are a few challenges we face with dimensionality reduction techniques which are as follows:

1. The new components created by PCA/SVD are **less interpretable**. They are the combination of the independent variables in the dataset and do not actually relate to the real-world, hence it becomes quite difficult to relate them to the real-world scenario.
2. **Numeric variables** are required for PCA/SVD. And hence all the categorical variables have to be represented in numeric form.
3. **Normalization/standardization** of the dataset is required to be done before the solution can be implemented.
4. There might be **information loss** when we use PCA or SVD. The principal components *cannot* replace the original dataset and hence there might be some loss of information while we implement these methods.

But despite a few challenges, PCA and SVD are used for reducing the dimensions in the dataset. They are one of the most popular methods and

quite heavily used. At the same time, it is imperative to note that

With this we have covered the two important techniques used in dimensionality reduction. We will examine more advanced techniques in the later chapters. It is time to move to the case study which is the next section of the chapter.

3.8 Case study for dimension reduction

We will now explore a real-world case to relate usage of PCA and SVD in real-world business scenarios.

Consider this. You are working for a telecommunication service provider like Vodafone, Verizon etc. You have a subscriber base, and you wish to cluster the consumers over a number of parameters. But the challenge can be the huge number of dimensions available to be analyzed.

The objective will be to reduce the number of attributes using dimension reduction algorithms. The consumer dataset can look like below.

1. Demographic details of the subscriber will consist of age, gender, occupation, household size, marital status etc. The list shown below is not exhaustive.

Table 3-8 Demographic details of a subscriber like age, gender, marital status, household size, City etc.

Mobile Number	Age	Gender	Marital Status	Household size	City	Country	Children	...
12345678900	25	M	Y	2	New York	US	0	
98765432100	26	F	N	4	London	UK	0	
45656465210	27	U	Y	4	New Delhi	India	2	
89323242111	28	M	N	2	Dublin	Ireland	0	
31822338924	29	F	Y	5	Tokyo	Japan	3	

2. Subscription details of the consumer might look like the table below. The list shown below is not exhaustive.

Table 3-9 Subscription details of a subscriber like tenure, postpaid/prepaid connection etc.

Mobile Number	Prepaid/Postpaid	Tenure	Homebroadband included	Family pack included	...
12345678900	Prepaid	1	Y	N	
98765432100	Postpaid	1.5	Y	N	
45656465210	Prepaid	1.2	N	Y	
89323242111	Prepaid	2	Y	Y	
31822338924	Postpaid	5	N	Y	

3. The usage of the consumer will describe the minutes, call rates, data usages, services etc. The list shown below is not exhaustive.

Table 3-10 Usage of a subscriber specifies the number of minutes used, SMS sent, data used, days spent in a network, national or international usage etc.

Mobile Number	Minutes	SMS	Data usage	National minutes	Days on network	International minutes	National SMS	International SMS	.....
12345678900	199	123	1 GB	170	24	101	104	141	
98765432100	105	119	2 GB	118	10	120	116	123	
45656465210	130	137	2.5 GB	156	23	181	182	181	
89323242111	110	161	4 GB	162	18	125	116	157	
31822338924	186	172	5 GB	139	25	177	167	138	

4. Payment and transaction details of the subscribers will talk about the various transactions made, the mode of payment, frequency of payments, days since last payment made etc.

Table 3-11 Transaction details of a subscriber showing all the details of amount, mode etc.

Mobile Number	# of transactions	Value	Mode	Frequency	...
12345678900	20	100	Cash	Monthly	
98765432100	15	150	Card	Yearly	
45656465210	25	1000	Online	Monthly	
89323242111	5	10	Voucher	Monthly	
31822338924	40	400	Cash	Weekly	

The dataset can have many more attributes. So far, we have established that the number of datasets involved are indeed high. Once we join all these

datapoints, the number of dimensions in the final data can be very huge.

Table 3-12 The final dataset is a combination of all the above-mentioned datasets. It will be a big, really high-dimensional dataset which is to be analyzed

Mobile Number	Age	Gender	Marital Status	Household size	City	Country	Children	Prepaid/Post	Tenure	Homebroadband included	Family pack	Minutes	SMS	Data usage	National minutes	Days on network	International minutes	National SMS	International SMS	# of transact	Value	Mode	Frequency
1234567890	25	M	Y	2	New York	US	0	Prepaid	1	Y	N	139	158	1 GB	151	13	138	147	145	20	100	Cash	Monthly
9876543210	26	F	N	4	London	UK	0	Postpaid	1.5	Y	N	199	200	2 GB	127	27	162	148	184	15	150	Card	Yearly
4505640521	27	U	Y	4	New Delhi	India	3	Prepaid	1.2	N	Y	185	155	2.5 GB	139	12	170	191	146	25	1000	Online	Monthly
8932324211	28	M	N	2	Dublin	Ireland	0	Prepaid	2	Y	Y	102	146	4 GB	173	27	195	166	182	5	10	Voucher	Monthly
3182233802	29	F	Y	5	Tokyo	Japan	3	Postpaid	5	N	Y	200	151	5 GB	146	2	101	162	199	40	400	Cash	Weekly

We have to reduce the number of attributes before we can proceed to any supervised or unsupervised solution. In this chapter, we are focusing on dimensionality reduction techniques and hence the steps are covering that aspect of the process. In the later chapters, we will examine the exploratory analysis in more detail.

As a first step, we will perform a sanity check of the dataset and do the data cleaning. We will examine the number of data points, number of missing values, duplicates, junk values present etc. This will allow us to delete any variables which might be very sparse or contain not much information. For example, if the gender is available for only 0.01% of the population it might be a good idea to drop the variable. Or if all the customers have gender as male, the variable is not adding any new information to us and hence it can be discarded. Sometimes, using business logic a variable might be dropped from the dataset. An example has been discussed in the earlier sections. In this step, we might combine a few variables. For example, we might create a new variable as average transaction value by dividing total amount spent by total number of transactions. In this way, we will be able to reduce a few dimensions.

A Python Jupyter notebook is checked-in at the Github repository, wherein we have given a very detailed solution for data cleaning step.

Once we are done with the basic cleaning of the data, we start with the exploratory data analysis. As a part of exploratory analysis, we examine the spread of the variable, its distribution, mean/median/mode of numeric variables. This is sometimes referred to as *univariate analysis*. This step allows us to measure the spread of the variables, understand the central tendencies, examine the distribution of different classes for categorical

variables and look for any anomalies in the values. For example, using the dataset mentioned above we will be interested to analyse the maximum/minimum/average data usage or the % distribution of gender or age. We would want to know the most popular method to make a transaction and we would also be interested to know maximum/minimum/average amount of the transactions. And this list will go on.

Then we explore the relationships between variables which is referred to as *bivariate analysis*. Crosstabs, distribution of data is a part of bivariate analysis. Correlation matrix is created during this step. Variables which are highly correlated are examined thoroughly. And based on a business logic, one of them might be dropped. This step is useful to visualize and understand the behavior of one variable in the presence of other variables. We can examine their mutual relationships and respective strength of the relationships. In this case study, we would answer the questions like – do subscribers who use more data spend more time on network as compared to subscribers who send more SMS. Or hypothesis like – do the subscribers who make a transaction using online mode generate more revenue than the ones using cash. Or is there a relationship between gender/age with the data usage. Many such questions are answered during this phase of the project.

A Python Jupyter notebook is checked-in at the Github repository, which provides detailed steps and code for the univariate and bivariate phase. Check it out!

At this step, we have a dataset which has a huge number of dimensions and we require to reduce the number of dimensions. Now it is good time to implement PCA or SVD. The techniques will reduce the number of dimensions and will make the dataset ready for the next steps in the process, as shown in (Figure 3-9). The figure is only representative in nature to depict the impact of dimensionality reduction methods. Notice how the large number of black lines in the left figure are getting reduced to lesser number of red lines in the right figure.

Figure 3-9 A very high dimensional dataset will be reduced to a low dimensional one by using principal components which capture the maximum variance in the dataset
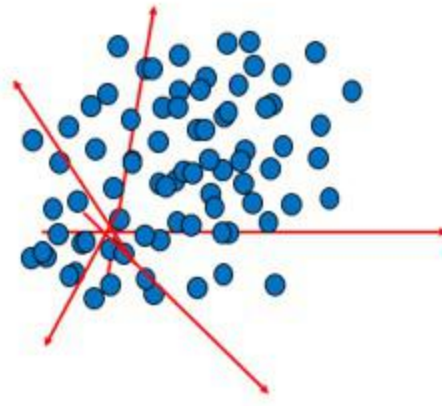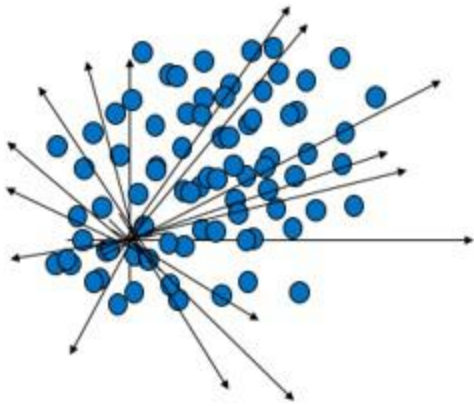
The output of dimensionality reduction methods will be a dataset with lower number of variables. The dataset can be then used for supervised or unsupervised learning. We have already looked at the examples using Python in the earlier sections of the chapter.

This concludes our case study on telecom subscribers. The case can be extended to any other domain like retail, BFSI, aviation, healthcare, manufacturing etc.

We will now proceed to the summary of the chapter.

3.9 Summary

Data is everywhere, in various forms, levels, dimensions and with varying levels of complexity. It is often mentioned that "the more data the better. It is indeed true to a certain extent. But with a really high number of dimensions, it becomes quite a herculean task to make sense out of it. The analysis can become biased and really complex to deal with. We explored this curse of dimensionality in this second chapter. PCA/SVD are helpful to reduce this complexity. They make the dataset ready for the next steps.

But dimensionality reduction is not as straightforward as it looks like. It is not an easy task. But it is certainly a very rewarding one. And requires a combination of business acumen, logic and common sense to deal with. The resultant dataset might still require some additional work. But it is a very good point for building a machine learning model.

This marks the end of third chapter. It also ends the part one of the book. In this part, we have covered the more basic algorithms. We started with the first chapter of the book, where we explored the fundamentals and basics of machine learning. In the second chapter we examined three algorithms for clustering. In this third chapter, we explored PCA and SVD.

In the second part of the book, we are changing gears and studying more advanced topics. We are starting with association rules in the next chapter. Then we go into advanced clustering methods of time-series clustering, fuzzy clustering, GMM clustering etc. It is followed by a chapter on advanced dimensionality reduction algorithms like t-SNE, LDA. And then to conclude the second part, we are examining unsupervised learning on text datasets. The third part of the book is even more advanced where we dive into neural network based solutions and use images datasets. So still a long way to go! Stay tuned!

You can proceed to the question section now!

Practical next steps and suggested readings

1. Use the vehicles dataset used in the last chapter for clustering and implement PCA and SVD on it. Compare the performance on clustering before and after implementing PCA and SVD.
2. Get the datasets from the (). Here, you will find a lot of datasets like USDA National Nutrient database, Pizza dataset etc. Compare the performance of PCA and SVD on these datasets.
3. Go through the following papers on PCA
4. https://www.sciencedirect.com/science/article/pii/009830049390090R
5. https://web.stanford.edu/~hastie/Papers/spc_jcgs.pdf
6. https://web.cs.ucdavis.edu/~vemuri/papers/pcaVisualization.pdf
7. https://cseweb.ucsd.edu/~ravir/papers/pca/pamifinal.pdf
8. Go through the following research papers on SVD
9. https://people.maths.ox.ac.uk/porterm/papers/s4.pdf
10. https://papers.nips.cc/paper/3473-quic-svd-fast-svd-using-cosine-trees.pdf
11. https://arxiv.org/pdf/1211.7102.pdf
12. http://glaros.dtc.umn.edu/gkhome/fetch/papers/sarwar_SVD.pdf

4 Association rules

"The power of association is stronger than the power of beauty; therefore the power of association is the power of beauty– John Ruskin"

Congratulations on finishing the first part of the book! You explored the basics of unsupervised learning and algorithms like k-means clustering, hierarchical clustering, DBSCAN, principal component analysis and others. It is expected that you have covered the mathematical concepts in the first part and created the Python codes to solve the exercise given at the end of each chapters.

Welcome to the second part of the book wherein we leverage the concepts learned in the first part and explore slightly more complex topics. We are starting with association rules in the chapter 4 of this book. All the very best!

Next time you visit a nearby grocery store, look around inside the store and the arrangements of various items. You would find shelves with items like milk, eggs, bread, sugar, washing powder, soaps, fruits, vegetables, cookies and various other items neatly stacked. Have you ever wondered what is the logic of this arrangement and how these items are laid out? Why certain products are kept near to each other while others are quite far from each other? Obviously, the arrangement cannot be done in a random manner and there has to be scientific reasoning behind it. Or do you wonder, how does Netflix recommend movies to you based on your movie history like a sequence? We are going to find the answers to these questions in this chapter. Like always, we will study the concepts first. We will go through the mathematical logic for different algorithms, pros and cons of each and practical implementations using Python. A business case study is provided at the end of the chapter to complement the knowledge.

In this fourth chapter of the book, we are going to cover the following topics:

1. Association rule learning
2. Different types of association rules algorithms

Welcome to the fourth chapter and all the very best!

4.1 Technical toolkit

We will continue to use the same version of Python and Jupyter notebook as we have used so far. The codes and datasets used in this chapter have been checked-in at this location.

You would need to install a few Python libraries in this chapter which are – apyori, pyECLAT, fpgrowth_py and pyspade. Along with this we will need numpy and pandas. Using libraries, we can implement the algorithms very quickly. Otherwise, coding these algorithms is quite a time-consuming and painstaking task.

Let's get started with association rules.

4.2 Association rule learning

You might have heard the famous "*beer and diaper story*". As per this anecdote, customers (mostly young men) of a supermarket who buy diapers also buy beer in the same invoice. In other words, young men who are buying diapers for their babies have quite a high probability to buy beers in the same transaction. We will not comment on the authenticity of the story, but *association rule learning* can be attributed as the logic derived from this story.

Formally put - association rules can be used to find compelling relationships between the variables which are present in the data sets. We can use association rules for measuring the correlations and co-occurrences between the variables in a dataset. In the example given above (assuming the story is true), one could analyse the daily customer transactions. And if a relationship emerges between beer and diapers, it is a very strong insight for the

supermarket, which can allow them to customize their placements of beer and diapers or tailor the marketing strategy or even alter the prices.

We can understand by a different example in a supermarket. Consider the example below. Assume that by analysing five invoices generated in a supermarket, we get the data below as shown in Table 4-1. In this example, in invoice number 1001 milk is purchased hence it has value as 1, whereas cheese is not purchased, hence it is 0.

Table 4-1 Examples of invoices generated in a supermarket. The first invoice number is 1001 and, in that invoice, milk is bought. Hence, there is 1 in front of milk. While cheese is not bought in 1001 and hence, there is 0 in front of cheese.

Invoice Number	Milk	Eggs	Bread	Cheese
1001	1	1	1	0
1002	0	0	0	1
1003	1	1	1	0
1004	0	1	0	1
1005	1	1	0	1

So, in invoice number 1001, milk, eggs, bread are purchased while in invoice number 1002 only cheese is purchased. Here, we can see that whenever milk and eggs are purchased together, bread is always purchased in the same invoice. It is an important discovery indeed.

Now scale up this understanding to thousands of transactions made in a day.

It will lead to very strong relationships which are generally oblivious to human eyes. But association rule algorithms can uncover them for us. It can lead to better product placements, better prices of the products and much more optimized marketing spends. Such patterns will enhance the customer experience and prove quite handy to improve the overall customer satisfaction.

We can visualize association rules as shown in (Figure 4-1). Here, there are some incoming variables represented as nodes 1,2,3,4 etc. These nodes are related to each other as shown by the arrows. This relationship between them gives rise to rules A and B.

Figure 4-1 Association rule can be visualized as the relationships between various variables in the dataset. These variables are linked to each other, and significant relationships are established between them.
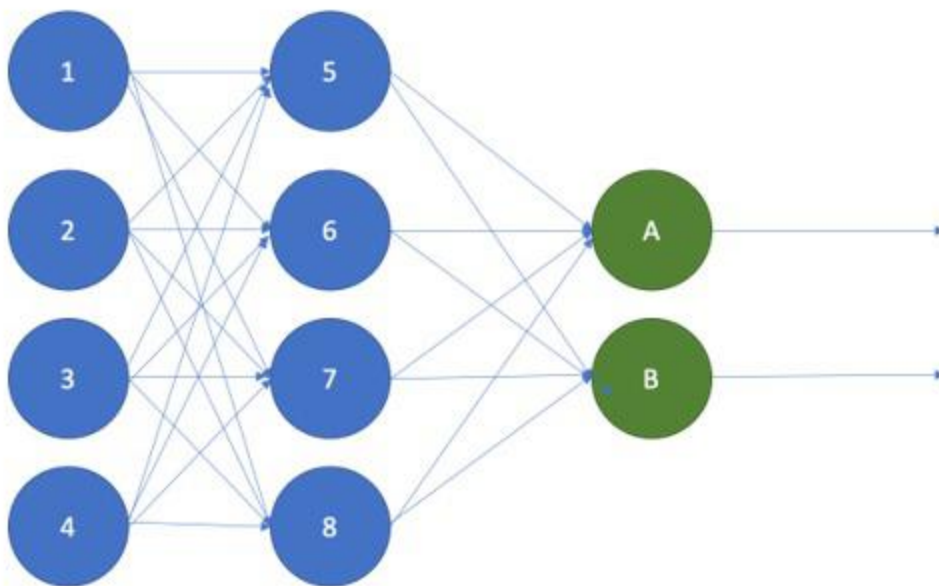
The example of the supermarket discussed above is sometime referred to as *Market Basket Analysis.* But association rules are applicable not only in grocery retail. Their utility has been proven in other sectors like bioinformatics and medical industry, intrusion detection etc. They can be utilised by Netflix or Spotify to analyse the historical user behaviour and then recommend the content which the user most likely is going to like. Web developers can analyse the historical clicks and usages of the customers on

their websites. By identifying the patterns, they can find out what user tends to click and which features will maximise their engagements. Medical practitioners can use association rules to better diagnose patients. The doctors can compare the probability of the symptoms in relationships with other symptoms and provide more accurate diagnosis. The use cases are across multiple business domains and business functions.

We will now understand the building blocks for association rules.

4.3 Building blocks of association rule

We covered the definition of association rule in the last section.

Now let's understand the mathematical concept behind association rules. Assume that we have the following datasets in a retail store-

1. Let X = {x, x_2, x_3, x_4, x_5 ….., x_n} ar`e the *n* items available in the retail store. For example, they can be milk, eggs, bread, cheese, apples and so on.
2. Let Y = {y, y_2, y_3, y_4, y_5 ….., y_m} are the *m* transactions generated in that retail store. Each transaction could have all or some items from the retail store.

Obviously, each item in the transactions will be bought from the retail store only. Or, in other words, every item in transactions in set Y will be a subset of items in the set X. At the same time, each item would have a unique identifier attached to it and each transaction would have a unique invoice number attached to it.

Now, we are interested to analyze the patterns and discover the relationships. This will be used to generate any rule or insight. So, let's define the meaning of rule first.

3. Assume that we find a rule that whenever items in a list P are bought, items in list Q are also bought. This rule can be written as follows:
4. The rule is **P -> Q**. It means that whenever items defined in P are bought it leads to the purchase of Q too.

5. Items in P will be a subset of Y or **P Í Y**.
6. Similarly, items in Q will be a subset of Y or **Q Í Y**.
7. P and Q cannot have any common element or **P Ç Q = 0**

Now, let's understand these mathematical concepts with a real-world example.

Assume that X = {milk, bananas, eggs, cheese, apples, bread, salt, sugar, cookies, butter, cold-drinks, water}. These are the total items available in the retail shop.

Y = {1001, 1002, 1003, 1004, 1005}. These are the five invoices generated in that retail store. The respective items purchased in each of these invoices is given in Table 4-2.

Table 4-2 Example of five invoices generated in a retail store. Note how for each invoice, we have 0 and 1 associated for each of the items. These invoices are just for illustration purpose. In the actual invoices, the number of items can be much more.

Invoice Number	Milk	Bananas	Eggs	Cheese	Apples	Bread	Salt	Sugar	Cookies	Butter	Colddrinks	Water
1001	1	1	1	0	0	0	1	0	0	1	1	1
1002	0	0	0	1	0	0	1	0	0	1	0	1
1003	1	1	1	0	1	0	0	0	1	0	0	1
1004	0	1	0	1	1	1	1	0	1	0	0	0
1005	1	1	1	1	0	0	0	1	0	1	1	0

Using this dataset, let's assume we create two rules that {milk, bananas} -> {eggs} and {milk, bananas} -> {bread}.

First rule means that whenever milk and bananas are bought together, eggs are also purchased in the same transaction. And second rule means that whenever milk and bananas are bought together, bread is also bought in the same transaction. By analyzing the dataset above, we can clearly see that rule 1 is always true whereas rule 2 is not.

The items on the left side are called the *antecedent* or the LHS and the ones on the right side are called the *consequents* or the RHS.

In the real-world, for any such rule to have significance, the same pattern must repeat itself across several hundreds and thousands transactions. Then

only we would conclude that the rule is indeed true and can be generalized across the entire data base.

At the same time, there can be many such rules. In a retail shop where daily thousands of invoices are generated, there can be hundreds of such rules. How can be find out which rules are significant, and which are not? This can be understood using the concept of support, confidence and lift which we will study in the next section.

4.3.1 Support, confidence, lift, and conviction

We identified the meaning of a rule in association rule mining. We also understand that there can be hundreds of rules based on the transactional data set. In this section we will explore, how we can measure the effectiveness of such rules and can short list the most interesting ones. This can be achieved using the concepts of support, confidence, lift, and conviction.

Recall in the last section we discussed about generalization of a rule. Support, confidence, lift, and conviction allow us to measure the level of generalization. In simple words, using these three parameters we can determine how useful the rule can be in our pragmatic real-world business. After all, if a rule is not useful or is not powerful enough, it is not required to be implemented. Support, confidence, lift, and conviction are the parameters to check the efficacy of the rule. We will understand these concepts in detail now.

We will use the below data set in Table 4-3 to understand the concept of support, confidence, and lift.

Table 4-3 Data set we are going to use to understand the concept of support, confidence, and lift. The first invoice 1001 has milk, eggs, and bread while cheese is not purchased. Again, for the sake of this example, we have taken only 4 items in total.

Invoice Number	Milk	Eggs	Bread	Cheese
1001	1	1	1	0

1002	0	1	1	1
1003	1	1	1	0
1004	0	1	0	1
1005	0	1	1	0

Here, for an invoice, 1 represents if an item is present in that invoice while 0 shows that the item was not purchased in that particular invoice. For example, invoice number 1001 has milk, eggs and bread while 1002 has eggs, bread and cheese.

Let's study support now.

Support

Support measures the frequency percentage of the items in the datasets. In other words, it measures the percentage of transactions in which the items are occurring in the data set.

Support can be denoted as shown below

$$Support = \frac{(total\ number\ of\ transactions\ in\ which\ the\ item\ of\ rule\ is\ present)}{(Total\ number\ of\ transactions\ present\ in\ the\ data\ base)}$$

Refer to Table 4-3, if we are interested in the rule {milk, eggs} -> {bread}. In such a scenario, there are two transactions in which all three items (milk, eggs, and bread) are present. The total number of transactions are five. So, it means that the support for the rule is 2 / 5 which is 0.4 or 40%.

If we are interested in the rule {bread, eggs} -> {cheese}. In such a scenario,

there is only one transaction in which all three items are present. The total number of transactions are five. So, it means that the support for the rule is 1 / 5 which is 0.2 or 20%.

Higher the support for a rule, better it is. Generally, we put a minimum threshold to get support. Minimum threshold is generally determined in consultation with the business stakeholders.

We will now study confidence for a rule.

Confidence

Confidence measures how often the rule is true. In other words, it measures the percentage of transactions which if contain antecedents also contain consequents.

So, if we wish to measure the confidence of the rule A->B

$$Confidence = \frac{support\ (A \cup B)}{support\ (A)}$$

Here, the numerator is support when both A and B are present in the transaction, while the denominator refers to the support only for A.

Refer to Table 4-3, if we are interested in the rule {milk, eggs} -> {bread}. In such a scenario, there are two transactions in which both milk and eggs are present. Hence, the support is 2/5 = 0.4. It is the denominator. There are two transactions in which all three (milk, eggs, bread) are present. Hence, support is 2/5 = 0.4, which is the numerator. Putting in the equation above, the confidence for the rule {milk, eggs} -> {bread} is 0.4/0.4 = 1.

If we are interested in the rule {eggs, bread} -> {cheese}. In such a scenario, there are three transactions in which (eggs, bread) are present. The total number of transactions are five. So, it means that the support is 3 / 5 which is 0.6. There is only one transaction in which all the three items (eggs, bread, cheese) are present. So, the support is 1/5 = 0.2. Hence the confidence for the rule {eggs, bread} -> {cheese}is 0.2/0.6 = 0.33.

Higher the confidence for the rule, better it is. Like support, we put a minimum threshold on confidence.

Sometimes, it is also referred to as the *conditional probability* of A on B. It can be understood as probability of B occurring provided A has already occurred and can be written as P(A|B). So, in the examples quoted above, probability of cheese to be bought provided eggs, bread is already bought is 33% while probability of bread to be purchased, provided milk, eggs are already purchased is 100%

We have covered confidence and support so far. We will now study lift and conviction for a rule which are real criteria to evaluate a rule.

Lift and conviction

Lift is a very important measurement criteria for a rule. Lift for a rule A-> B can be defined as

$$Lift\ (A \rightarrow B) = \frac{support\ (A \cup B)}{support\ (A) * support\ (B)}$$

Here, the numerator is support when both A and B are present in the transaction, while the denominator refers to the support for A multiplied by support for B.

Refer to Table 4-3, if we are interested in the rule {milk, eggs} -> {bread}. In such a scenario, there are two transactions in which all three (milk, eggs, bread) are present. Hence, support is again 2/5 = 0.4, which is the numerator. There are two transactions in which only (milk, eggs) are present, so the support is 2/5 = 0.4. There are four transactions in which bread is present, hence the support is 4/5 = 0.8. Putting in the equation above, the lift for the rule {milk, eggs} -> {bread} is 0.4/(0.4x0.8) = 1.25.

If we are interested in the rule {eggs, bread} -> {cheese}. In such a scenario, there is only one transaction in which (eggs, bread, cheese) are present. The total number of transactions are five. So, it means that the support is 1 / 5 which is 0.2. There are two transactions in which (cheese) is present. So, the

support is 2/5 = 0.4. There are four transactions in which (eggs, bread) are present, so the support is 4/5 = 0.8. Putting in the equation above, the lift for the rule {eggs, bread} -> {cheese} is 0.2/(0.4x0.8) = 0.625.

If the value of the lift is *equal to 1*, it means that the antecedent and precedent are independent of each other, and no rule can be drawn from it.

If the value of lift is *greater than one*, it means that the antecedent and precedent are dependent on each other. This rule can be used for predicting the antecedent in future transactions. This is the insight we want to draw from the data set.

The value of lift is *lesser then one*, it means that the antecedent and precedent are substitute of each other. The presence of one can have a negative impact on the other. It is also an important insight which can be used by the business teams for strategic planning.

While we evaluate any rule using the lift, it is imperative that we apply domain knowledge to it. For example, if we evaluate the rule {eggs, bread} -> {cheese}, it suggests that eggs, bread can be a substitute for cheese. We know that it is not true in the real life. Hence, in such a scenario we cannot make any decision for this role. We must take help of domain knowledge to draw any conclusions for this rule.

At the same time, rule {milk, eggs} -> {bread} seems to be quite logical. We know that when milk and eggs are purchased together it is highly likely that bread will be purchased too in the same transaction. Hence this rule makes much more sense to us.

Conviction is an important parameter which is given by the formula below

$$Conviction\ (A \rightarrow B) = \frac{1 - support\ (B)}{1 - confidence\ (A \rightarrow B)}$$

For example, if we are interested in the rule {eggs, bread} -> {cheese}. In such a scenario, there is only one transaction in which (cheese) is present. The total number of transactions are five. So, it means that the support is 1 / 5 which is 0.2 and will be used in the numerator. We have already calculated

the confidence as 0.625. Putting back in the formula, we can calculate conviction as (1-0.2)/(1-0.625) = 2.13

We can interpret the conviction as – the rule {eggs, bread} -> {cheese} would be incorrect 2.13 times more often if the association between {eggs, bread, cheese} was purely chosen at random.

In most of the business scenarios, lift is the measurement criteria used. There are other measurement parameters too like leverage, collective strength etc. But most of the time confidence, support and lift are used to measure the effectiveness of any rule.

🎯 POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

1. Support measures how often the rule is present in the dataset. True or False

2. If the lift is greater than 1, it means that the two items are independent of each other. True or False.

3. Lower the value of confidence, better is the rule. True or False.

While we evaluate any rule while analyzing the data set, most of the time we set a threshold for the confidence, support, and the lift. It allows us to reduce the number of rules and filter out the irrelevant ones. In other words, we are interested in only the rules which are very frequent. We will study it in more detail when we create a Python solution for a dataset.

We will now study the various algorithms used in association rule. The first such algorithm is Apriori algorithm which is the next topic.

4.4 Apriori algorithm

Apriori algorithm is one of the most popular algorithms used for association

rules. It was proposed by Agrawal and Shrikant in 1994. The link to the paper is given at the end of the chapter.

Apriori is used to understand and analyse the frequent items in a transactional database. It utilizes a "bottoms-up" approach where first candidates are generated based of the frequency of the subsets. Let us understand the entire process by means of an example.

We will use the same dataset we have discussed earlier.

Table 4-3 Data set we are going to use to understand the concept of support, confidence, and lift. The first invoice 1001 has milk, eggs, and bread while cheese is not purchased.

Invoice Number	Milk	Eggs	Bread	Cheese
1001	1	1	1	0
1002	0	1	1	1
1003	1	1	1	0
1004	0	1	0	1
1005	0	1	1	0

The process used in Apriori algorithm it will look like the process below in Figure 4-2.

Figure 4-2 Apriori algorithm process can be depicted as shown here.
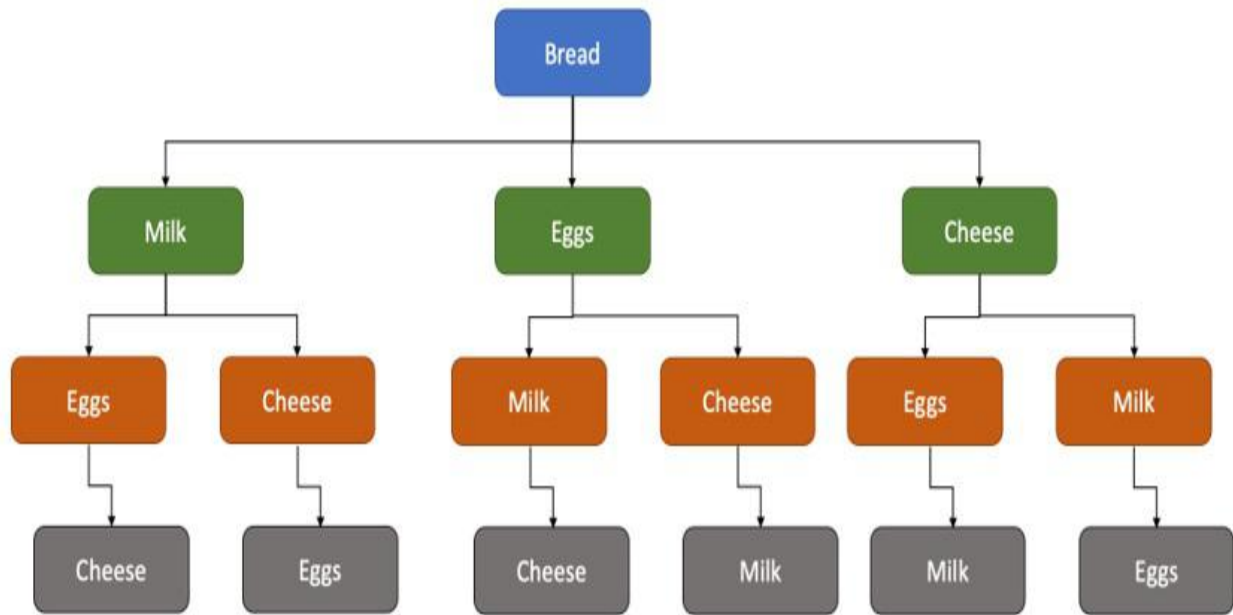
Let us say we wish to analyse the relationship of bread with all the other items in the dataset. In this case, the level 1 is Bread and we find its frequency of occurrence.

Then we move to the next layer which is Layer 2. Now we find the relationship of Bread with each of the other items – Milk, Eggs and Cheese which are at Layer 2. Here again we find the respective frequencies of occurrence for all the possible combinations which are {Bread, Milk}, {Bread, Eggs} and {Bread, Cheese}. It can be shown in Figure 4-3.

Figure 4-3 We have bread at Level 1 while the other items (milk, eggs, and cheese) are kept at Level 2. Bread is kept at level 1 since we wish to analyze the relationship of bread with all the other items.
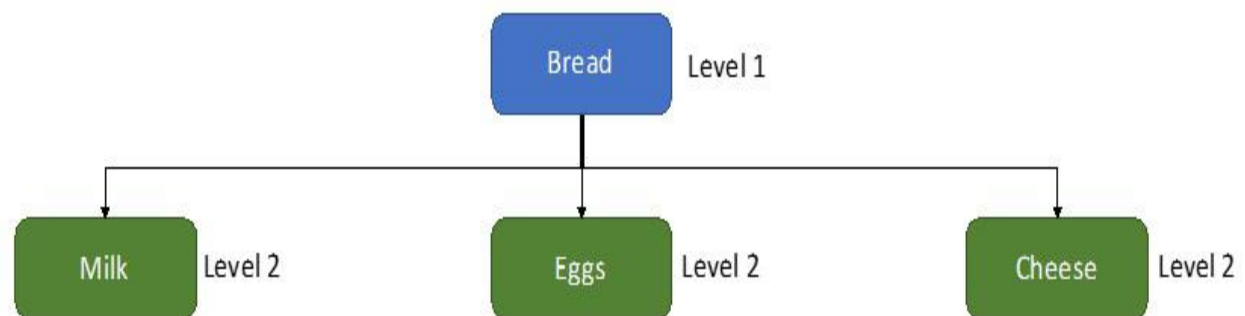
After this Layer 2 has been analysed, we move to the third layer and fourth

layer and so on. This process continues till we reach the last layer wherein all the items have been exhausted.

As a result of this process, we can calculate the support for all the possible combinations. For example, we would know the support for

{Bread} -> {Milk},

{Bread} -> {Eggs} and

{Bread} -> {Cheese}.

For the next level, we would also get the support for

{Bread, Milk} -> {Eggs},

{Bread, Eggs} -> {Milk},

{Bread, Milk} -> {Cheese},

{Bread, Cheese} -> {Milk},

{Bread, Cheese} -> {Eggs} and

{Bread, Eggs} -> {Cheese}.

Now, using the same process, all the possible combinations for the next level are calculated. For example, {Bread, Eggs, Milk} -> {Cheese}, {Bread, Eggs, Cheese} -> {Milk} and so on.

When all the item sets have exhausted, the process will stop. The complete architecture can look like in Figure 4-4.

Figure 4-4 The complete architecture for Apriori algorithm. Here, we would have calculated support for all the possible combinations. The relationships between all the items are explored and because of this entire database scan, the speed of Apriori gets hampered.
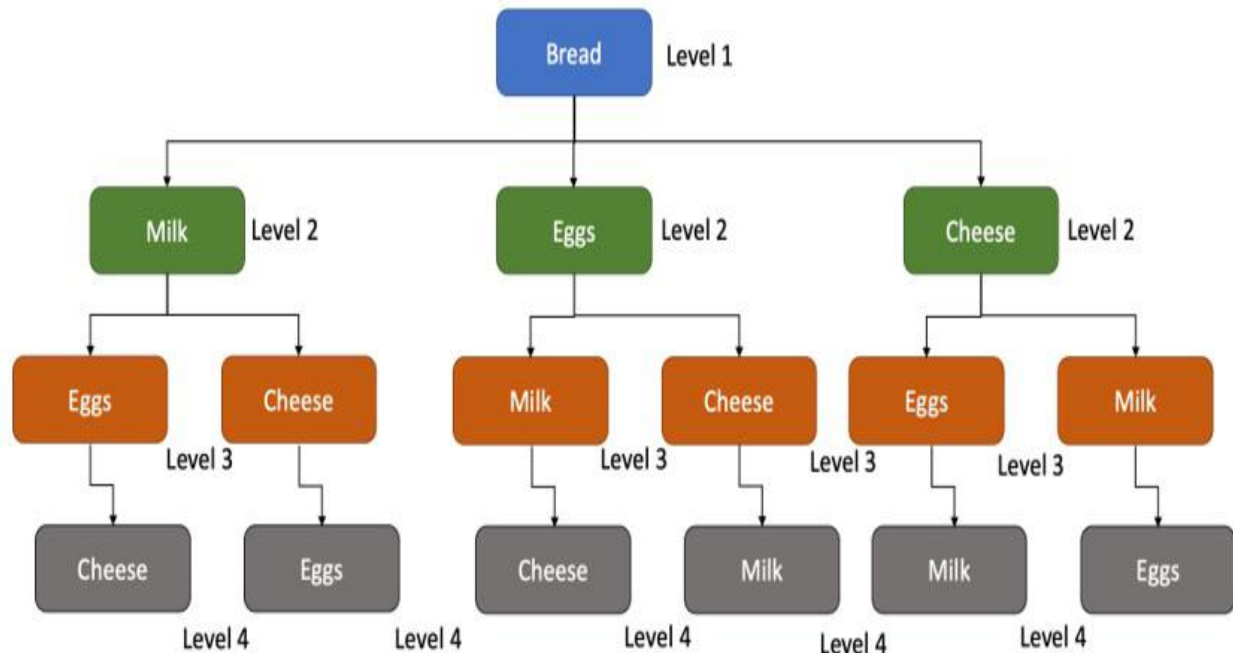
Now, we can easily understand that the possible number of combinations are quite high, which is one of the challenges with apriori. There are a few other shortcomings for Apriori algorithm which we will study later. But right now is the time to implement Apriori using Python.

4.4.1 Python implementation

We will now proceed with Python implementation of Apriori algorithm. The dataset and Python Jupyter notebook is checked-in at the GitHub repository.

You might have to install apyori.

To install the libraries, simply do the step below

```
import sys
!{sys.executable} -m pip install apyori
```

Step 1. Import the necessary libraries for the use case. We are importing numpy, pandas. For implementing apriori, we have a library called apyori which is also imported.

```
import numpy as np
import pandas as pd
```

```
from apyori import apriori
```

Step 2: We now import the datset store_data.csv file.

```
store_dataset = pd.read_csv('store_data.csv')
```

You are also advised to have a look at the dataset by opening the .csv file. It will look like the screenshot below. The first 25 rows are shown in the screenshot. Each row represents an invoice.

Step 3: Let's perform some basic checks on the data by `.info`, `.head` command.

```
store_dataset.info()
```

```
store_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7500 entries, 0 to 7499
Data columns (total 20 columns):
shrimp               7500 non-null object
almonds              5746 non-null object
avocado              4388 non-null object
vegetables mix       3344 non-null object
green grapes         2528 non-null object
whole weat flour     1863 non-null object
yams                 1368 non-null object
cottage cheese        980 non-null object
energy drink          653 non-null object
tomato juice          394 non-null object
low fat yogurt        255 non-null object
green tea             153 non-null object
honey                  86 non-null object
salad                  46 non-null object
mineral water          24 non-null object
salmon                  7 non-null object
antioxydant juice       3 non-null object
frozen smoothie         3 non-null object
spinach                 2 non-null object
olive oil               0 non-null float64
dtypes: float64(1), object(19)
memory usage: 1.1+ MB
```

```
store_dataset.head()
```

`store_dataset.head()`

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice
0	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Step 4: Here we can see that the first transaction has been considered as the header by the code. Hence, we would import the data again but this time he would specify that headers are equal to None.

```
store_dataset = pd.read_csv('store_data.csv', header=None)
```

Step 5: Let's look at the head again. This time it looks correct.

```
store_dataset.head()
```

```
store_dataset.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Step 6: The library we are using for the code accepts the dataset as a list of lists. The entire dataset must be a big list while each transaction is an inner list in the big list. So, to achieve it, we first convert our store_dataset data frame into a list.

```
all_records = []
for i in range(0, 7501):
    all_records.append([str(store_dataset.values[i,j]) for j in r
```

Step 7: Next, we implement the Apriori algorithm.

For the algorithm, we are working on all_records list we have created in Step 6. The minimum support specified is 0.5 or 50%, minimum confidence is 25%, minimum lift is 4 and minimum length of the rule is 2.

The output of this step is apriori_rules class object. This object is then converted into a list which we can understand. And finally, we print this list.

```
apriori_rules = apriori(all_records, min_support=0.5, min_confide
apriori_rules = list(apriori_rules)
print(len(apriori_rules))
```

The output of the code will be 0. It means that no such rules exist which satisfy the condition we have set for the rules.

We again try to execute the same code albeit by reducing the minimum support to 25%

```
apriori_rules = apriori(all_records, min_support=0.25, min_confid
apriori_rules = list(apriori_rules)
print(len(apriori_rules))
```

Again, no rules are generated and the output is zero. Even reducing the minimum support to 10% does not lead to any rules.

```
apriori_rules = apriori(all_records, min_support=0.1, min_confide
apriori_rules = list(apriori_rules)
print(len(apriori_rules))
```

Now, we reduce the minimum lift to 2. This time we get 200 as the output. It means that there are 200 such rules which fulfil the criteria.

```
apriori_rules = apriori(all_records, min_support=0.25, min_confid
apriori_rules = list(apriori_rules)
print(len(apriori_rules))
```

Step 8: Let's look at the first rule.

```
print(apriori_rules[0])
```

```
print(apriori_rules[0])

RelationRecord(items=frozenset({'almonds', 'burgers'}), support=0.005199306759098787, ordered_statistics=[OrderedStat
istic(items_base=frozenset({'almonds'}), items_add=frozenset({'burgers'}), confidence=0.25490196078431376, lift=2.923
577382023146)])
```

The rule explains the relationship between almonds and burgers. Support is .005 while the confidence is 0.25. Lift which is 2.92 indicates that this rule is quite strong in itself.

Step 9: We will now look at all the rules in detail. For that, loop through the rules and extract information from each of the iteration. Each of the rule has got the items constituting the rule and respective values for support, confidence and lift. We have shown an example in Step 8. Now in Step 9, we are just extracting that information for all of the rules using a for loop.

```
for rule in apriori_rules:
```

```
    item_pair = rule[0]
    items = [x for x in item_pair]
    print("The apriori rule is: " + items[0] + " -> " + items[1])

    print("The support for the rule is: " + str(rule[1]))

    print("The confidence for the rule is: " + str(rule[2][0][2])
    print("The lift for the rule is: " + str(rule[2][0][3]))
    print("***********************")
```

The output for this step is shown below. Here, we can observe the each rule in listed along with the respective values of support, confidence and lift.

```
for rule in apriori_rules:
    item_pair = rule[0]
    items = [x for x in item_pair]
    print("The apriori rule is: " + items[0] + " -> " + items[1])

    print("The support for the rule is: " + str(rule[1]))

    print("The confidence for the rule is: " + str(rule[2][0][2]))
    print("The lift for the rule is: " + str(rule[2][0][3]))
    print("***********************")
```

```
The apriori rule is: almonds -> burgers
The support for the rule is: 0.005199306759098787
The confidence for the rule is: 0.25490196078431376
The lift for the rule is: 2.923577382023146
***********************
The apriori rule is: cereals -> milk
The support for the rule is: 0.007065724570057326
The confidence for the rule is: 0.2746113989637306
The lift for the rule is: 2.119197637476279
***********************
The apriori rule is: chocolate -> tomato sauce
The support for the rule is: 0.005065991201173177
The confidence for the rule is: 0.3584905660377358
The lift for the rule is: 2.1879883936932925
***********************
The apriori rule is: mushroom cream sauce -> escalope
The support for the rule is: 0.005732568990801226
The confidence for the rule is: 0.3006993006993007
The lift for the rule is: 3.790832696715049
```

We can interpret the rules easily. For example, rule almonds-> burgers has a lift of 2.92 with a confidence of 25.49% and support of 0.51%. This concludes our implementation using Python. This example can be extended to any other real-world business dataset.

Not all the rules generated are not good for using. We will examine how to get the best rules from all the rules generated when we deal with the case study in the last section of the chapter.

Apriori algorithm is a robust and very insightful algorithm. But like any other

solution, it has a few shortcomings which we are discussing now.

4.4.2 Challenges with Apriori algorithm

We examined in previous sections how the number of subsets generated in Apriori algorithm are quite high.

Figure 4-5 Complete scan of dataset is done hence the speed is decreased significantly.
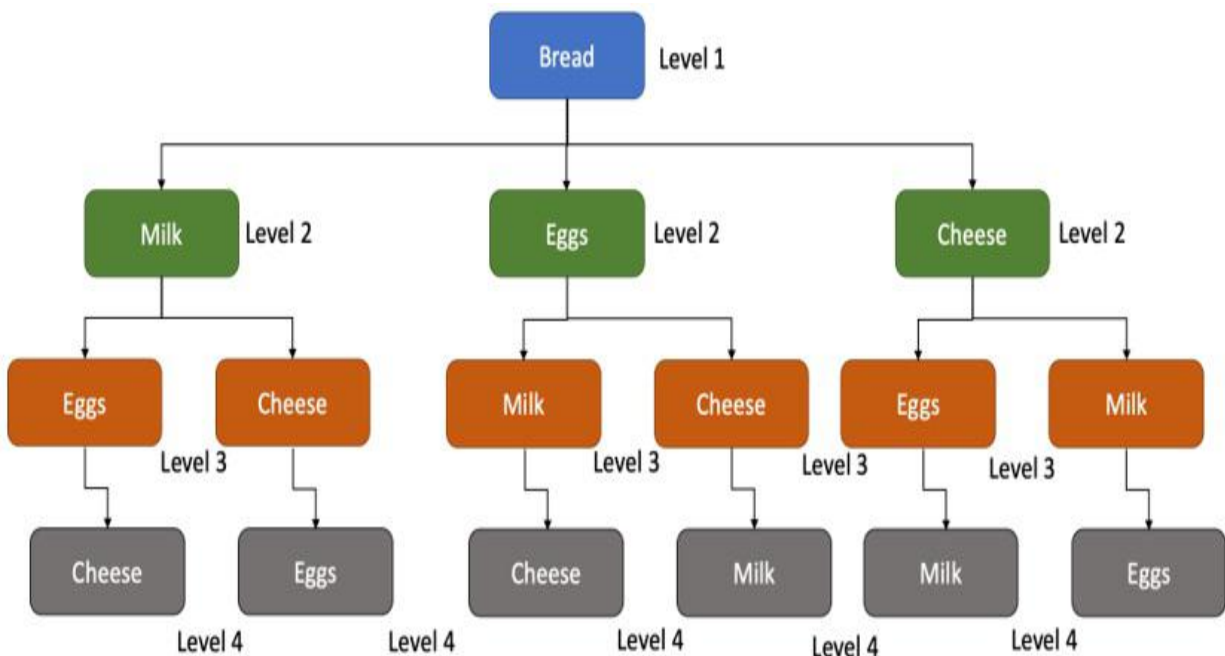
It is very tedious to generate candidates item sets and hence it becomes quite cumbersome to analyse the dataset. Apriori scans the entire dataset multiple times and hence it requires the database to be loaded in the memory. We can safely deduce that the complexity of the algorithm is quite high. It requires a lot of time to make the computations. This problem is magnified when we are dealing with a very large dataset. In fact, for real-world problems where millions of transactions are generated, quite a huge number of candidates itemsets are generated and hence it is very time consuming to use Apriori on the entire dataset.

Due to this very reason, generally a minimum value of support is set to reduce number of possible rules. In the example given above, we can calculate the support for level 1 combinations as shown below in Table 4-4.

Here, if we set the minimum value of support as 0.5, only one rule will be shortlisted.

Table 4-4 Support is calculated for each of the combination of the items. For example, for milk and bread – the number of transactions is 2 while the total number of transactions are 5. So, the support is 2/5 which is 0.4.

Combination	Number of txns	Total Txns	Support
Milk, Eggs	2	5	0.4
Milk, Bread	2	5	0.4
Milk, Cheese	0	5	0
Eggs, Bread	4	5	0.8
Eggs, Cheese	2	5	0.4
Bread, Cheese	1	5	0.2

Setting up a minimum value of support is hence an intelligent tactic to make the rules much more manageable. It reduces the time and generates the rules which are much more significant. After all, the rules generated from the analysis should be generalizable enough so that they can be implemented across the entire data base.

⌖ POP QUIZ – answer these question to check your understanding..

Answers at the end of the book

1. Apriori algorithm scans the database only once. TRUE or FALSE.

2. If bananas are present in 5 transactions out of a total of 12 transactions, it means the support for banana is 5/12. TRUE or FALSE.

But Apriori algorithm is a path-breaking solution. It is still highly popular and generally one of the very first algorithms whenever association rules are discussed.

Data preparation is one of the key steps and quite a challenge, we will explore this challenge during the case study in the last section of the chapter.

We will now study the next algorithm, which is ECLAT algorithm

4.5 Equivalence class clustering and bottom-up lattice traversal (ECLAT)

We will now study Equivalence class clustering and bottom-up lattice traversal algorithm or ECLAT in this section, which is sometimes said is better than apriori.

ECLAT uses a depth-first search approach. It means that ECLAT performs the search in a vertical fashion throughout the dataset. It starts at the root node. Then goes one level deep and continues delete reach the first terminal note. Let's say the terminal node is at level X. One start terminal node is reached, the algorithm then takes a step back and reaches level (X-1) and continues till it finds a terminal node again. Let's understand this process by means of a tree diagram as shown in Table 4-6.

Figure 4-6 Tree diagram to understand the process of ECLAT algorithm. It starts with 1 and ends at 16.
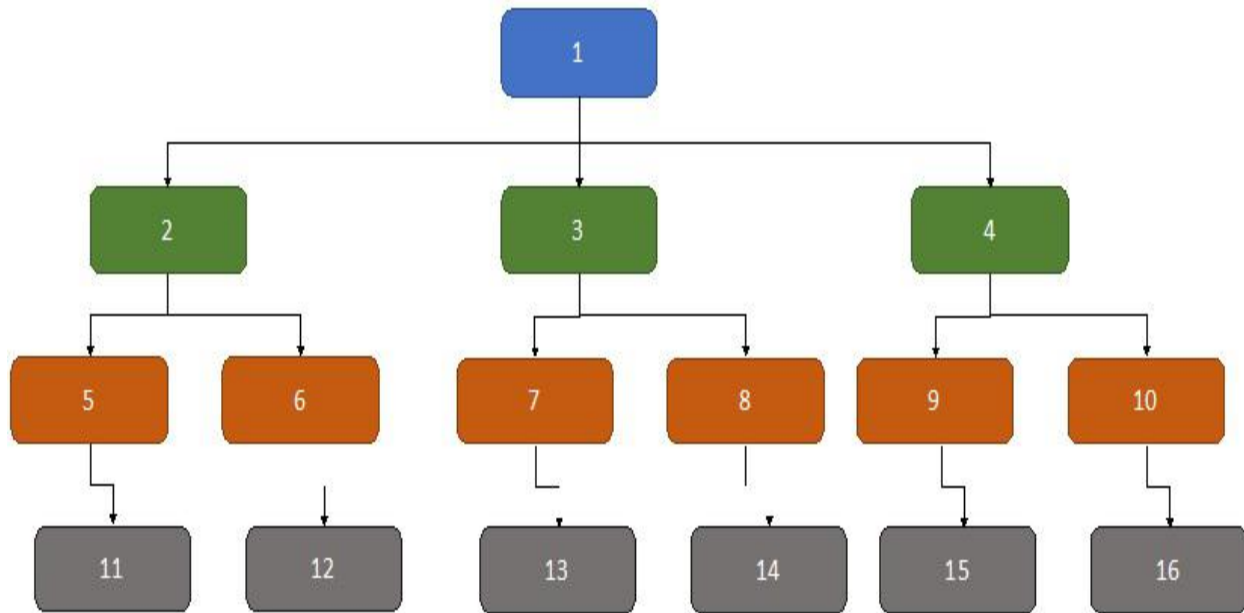
ECLAT will take the following steps:

1. The algorithm starts at the root node 1.
2. It then goes one level deep to root node 2.
3. It will then continue one more level deep till it reaches terminal node 11.
4. Once it reaches the terminal note 11, it then takes a step back and goes to node 5.
5. The algorithm then searches if there is any node available which can be used. At node 5 we can see that there is no such node available.
6. Hence the algorithm again takes a step back and it reaches node 2.
7. At node 2, the algorithm explores again. It finds that it is possible to go to note 6.
8. So, the algorithm goes to node 6 and starts exploring again till it reaches the terminal node 12.
9. This process continues till all the combinations have been exhausted.

Obviously, the speed of computation depends on the total number of distinct items present in the data set. It is because the number of distinct items define the width of the tree. The items purchased in each of the transactions would define the relationship between each node.

During execution time of ECLAT, each item (either individually or in a pair) is analyzed. Let us use the same example we have used for Apriori to

understand ECLAT better as shown in Table 4-5.

Table 4-5 Data set we are going to use to understand ECLAT. The first invoice 1001 has milk, eggs, and bread while cheese is not purchased.

Invoice Number	Milk	Eggs	Bread	Cheese
1001	1	1	1	0
1002	0	1	1	1
1003	1	1	1	0
1004	0	1	0	1
1005	0	1	1	0

ECLAT will undergo the following steps to analyze the dataset:

1. In the first run ECLAT will find the invoice numbers for all single items. Or in other words, it would find the invoice numbers for all the items individually. It can be shown in the Table 4-6 below, wherein milk is present in invoice number 1001 and 1003 while eggs are present in all the invoices.

Table 4-6 Respective invoices in which each item is present. Milk is present in 1001 and 1003 while eggs is present in five invoices.

Item	Invoice Numbers

Milk   1001,1003

Eggs   1001, 1002, 1003, 1004, 1005

Bread   1001, 1002, 1003, 1005

Cheese 1002, 1004

2. Now in the next step, all the two items dataset are explored as shown below in Table 4-7. For example, milk and eggs are present in invoice number 1001 and 1003, while milk and cheese are not present in any invoice.

Table 4-7 Two item data sets are explored now. Milk and eggs are present in invoice number 1001 and 1003 while there is no invoice for milk and cheese.

Item	Invoice Numbers
Milk, Eggs	1001, 1003
Milk, Bread	1001, 1003
Milk, Cheese	-
Eggs, Bread	1001, 1002, 1003, 1005
Eggs, Cheese	1002, 1004

Bread, Cheese 1002

3. In the next step, all the three item datasets are explored as shown in the table 4-8.

Table 4-8 Three item datasets are analyzed in this step. We have two combinations only.

Item	Invoice Numbers
Milk, Eggs, Bread	1001, 1003
Eggs, Bread, Cheese	1002

4. There are no invoices present in our data set which contains four items.
5. Now depending on the threshold, we set for the value of support count, we can choose the rules. So, if we want that minimum number of transactions in which the rule should be true is equal to three then only one rule qualifies which is {Eggs, Bread}. If we decide the threshold for the minimum number of transactions as two, then rules like {Milk, Eggs, Bread}, {Milk, Eggs}, {Milk, Bread}, {Eggs, Bread} and {Eggs, Cheese} qualify as the rules.

We will now create a Python solution for ECLAT.

4.5.1 Python implementation

We will now implement ECLAT using Python. The code is quite simple to understand. We are using pyECLAT library here. The dataset looks like

	A	B	C	D	E	F	G
1	shrimp	almonds	avocado	vegetables n	green grapes	whole weat	yams
2	burgers	meatballs	eggs				
3	chutney						
4	turkey	avocado					
5	mineral wate	milk	energy bar	whole wheat	green tea		
6	low fat yogurt						
7	whole wheat	french fries					
8	soup	light cream	shallot				
9	frozen veget	spaghetti	green tea				
10	french fries						
11	eggs	pet food					
12	cookies						
13	turkey	burgers	mineral wate	eggs		cooking oil	
14	spaghetti	champagne	cookies				
15	mineral wate	salmon					
16	mineral water						
17	shrimp	chocolate	chicken	honey	oil	cooking oil	low fat yogurt
18	turkey	eggs					
19	turkey	fresh tuna	tomatoes	spaghetti	mineral wate	black tea	salmon
20	meatballs	milk	honey	french fries	protein bar		
21	red wine	shrimp	pasta	pepper	eggs	chocolate	shampoo
22	rice	sparkling water					
23	spaghetti	mineral wate	ham	body spray	pancakes	green tea	
24	burgers	grated chees	shrimp	pasta	avocado	honey	white wine
25	eggs						
26	parmesan ch	spaghetti	soup	avocado	milk	fresh bread	

Step 1: We will import the libraries here.

```
import numpy as np
import pandas as pd
from pyECLAT import ECLAT
```

Step 2: Import the dataset now

```
data_frame = pd.read_csv('Data_ECLAT.csv', header = None)
```

Step 3: Here we are generating an ECLAT instance.

```
eclat = ECLAT(data=data_frame)
```

There are some properties of ECLAT instance `eclat` generated in the last step like eclat.df_bin which is a binary dataframe and eclat.uniq_ which is a list of all the unique itesms.

Step 4: We will now fit the model. We are giving a minimum support of 0.02 here. After that we are printing the support.

```
get_ECLAT_indexes, get_ECLAT_supports = eclat.fit(min_support=0.0
                                                    min_co
                                                    max_co
                                                    separa
get_ECLAT_supports
```

The output is

We can easily interpret the results here. For each of the item and combination of items, we are getting the value of the support. For example, for French fries and eggs, the value of support is 3.43%.

ECLAT has some advantages over Apriori algorithm. Since it uses a depth-search approach it is surely faster than Apriori and requires lesser memory to compute. It does not scan the dataset iteratively and hence it makes it even faster than Apriori. We will compare these algorithms once more after we have studied the last algorithm.

We will now move to the third algorithm which is F-P growth algorithm.

4.6 Frequent-Pattern growth algorithm (F-P algorithm)

F-P algorithm or frequent-pattern growth algorithm is the third algorithm we will discuss in this chapter. It is an improvement over the Apriori algorithm. Recall in Apriori we face challenges of time consuming and costly computations. FP resolves these issues by representing the data base in the form of a tree called a *frequent pattern tree or FP tree*. Because of this

frequent pattern, there is no need for generating the candidates as done in Apriori algorithm. Let's discuss FP in detail now.

FP tree or Frequent Pattern tree is a tree-shaped structure, and it mines the most frequent items in the datasets. It can look like Figure 4-7.

Figure 4-7 FP algorithm can be depicted in a tree-diagram structure. We will be creating this tree in a step-by-step fashion in the next steps. Each node represents a unique item. The root node is NULL.
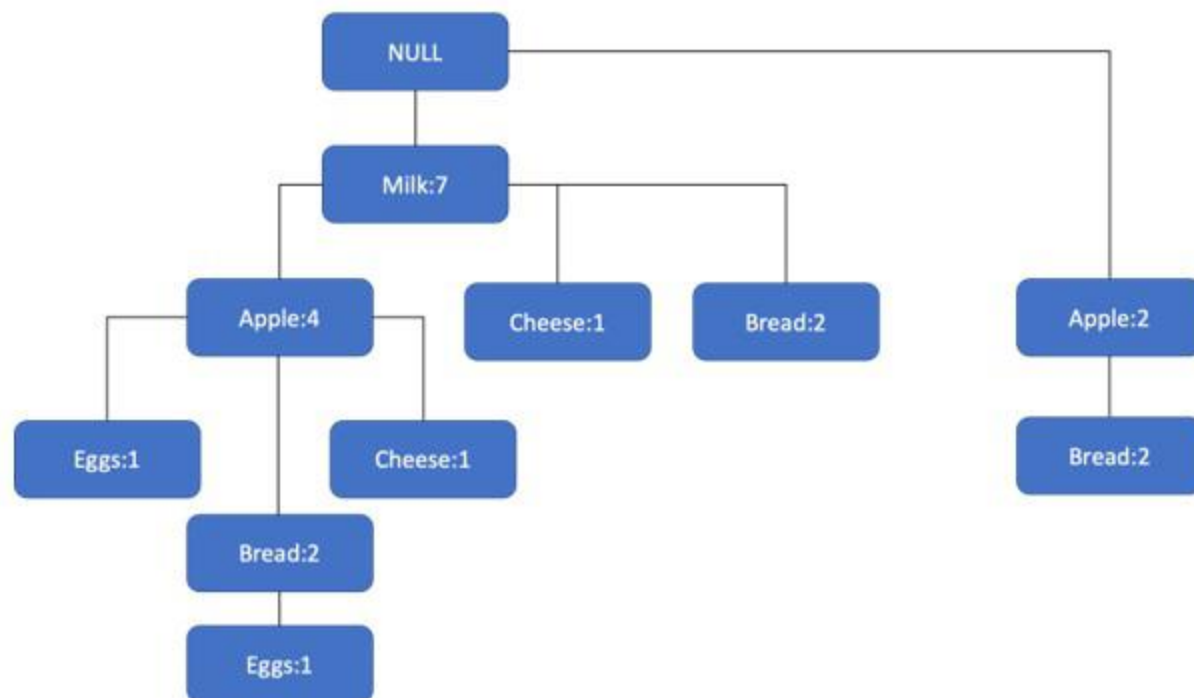
Each node represents the unique item in the dataset. The root node of the tree is generally kept as NULL. The other nodes in the tree are the items in the dataset. The nodes are connected with each other if they are in the same invoice. We will study the entire process in a step-by-step fashion.

Assume we are using the following dataset as shown in (Table 4-9). So, we have unique items as Apple, Milk, Eggs, Cheese and Bread. There are in total 9 transactions and the respective items in each of the transaction is shown in Table 4-9.

Table 4-9 Data set we are going to use to understand the concept FP algorithm. We have nine transactions here, for example in T1 we have apple, milk, and eggs.

Transactions Item sets

T1   Apple, Milk, Eggs

T2   Milk, Cheese

T3   Milk, Bread

T4   Apple, Milk, Cheese

T5   Apple, Bread

T6   Milk, Bread

T7   Apple, Bread

T8   Apple, Milk, Bread, Eggs

T9   Apple, Milk, Bread

Let's apply FP algorithm on this dataset now.

Step 1: Like Apriori, the entire dataset is scanned first. Occurrences for each of the items is counted and a frequency is generated. The results are as in Table 4-10. We have arranged the items in descending order of the frequency or the respective support count in the entire dataset.

Table 4-10 Respective frequency for each of the item set. For example, apples have been purchased in six transactions.

Item	Frequency or Support Count
Milk	7
Apple	6
Bread	6
Cheese	2
Eggs	2

If two items have exactly same frequency, anyone can be ordered first. In the example above, we have bread and apple having same frequency. So, we can keep either bread or apple as the first one.

Step 2: Let's start the construction of the FP tree. We start with creating the root node which is generally the NULL node in Figure 4-8.

Figure 4-8 The root node for the tree is generally kept NULL.

Step 3: Now analyse the first transaction T1. Here, we have Apple, Milk and Eggs in the first transaction. Out of these three, milk has the highest support count which is 7. So, a connection is extended from root node to Milk and we

denote it by Milk:1. We have shown in Figure 4-9.

Figure 4-9 Connection from the root node to Milk. Milk has the highest support hence we have chosen milk.
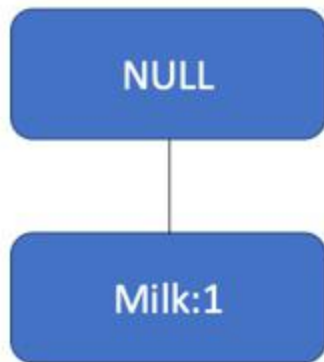
Step 4: We will now look at the other items in T1. Apple have a support count of 6 and Eggs have a support count of 2. So, we will extend the connection from Milk to Apple and name it Apple:1 and then from Apple to Eggs and call it Eggs:1. We have shown in Figure 4-10.

Figure 4-10 Step 4 of the process where we have finished all the items in T1. All the items milk, apple and eggs are now connected with each other.
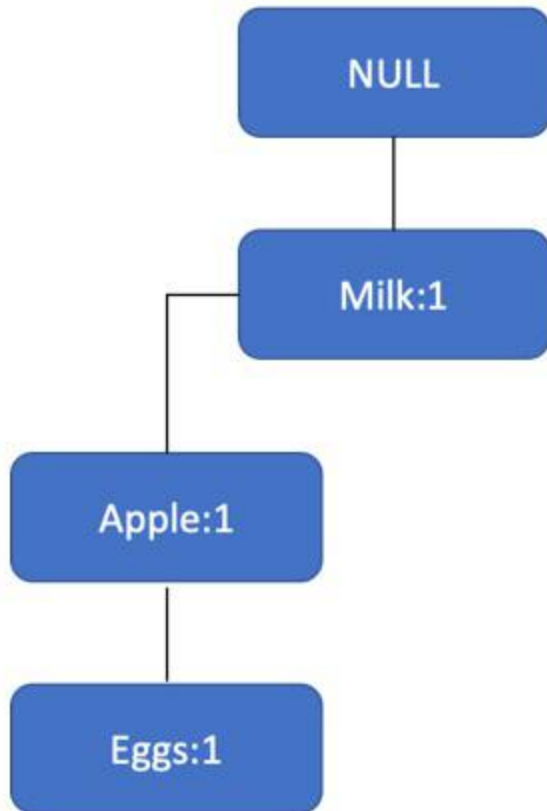
Step 5: Let's look at T2 now. It has Milk and Cheese. Milk is already connected to the root node. So, the count for Milk becomes 2 and it becomes Milk:2. We will next create a branch from Milk to cheese and name it Cheese:1. The addition is shown in Figure 4-11.

Figure 4-11 Step 5 of the process where we started to analyze T2. Milk is already connected so it's count increases by 2 while cheese gets added to the tree.
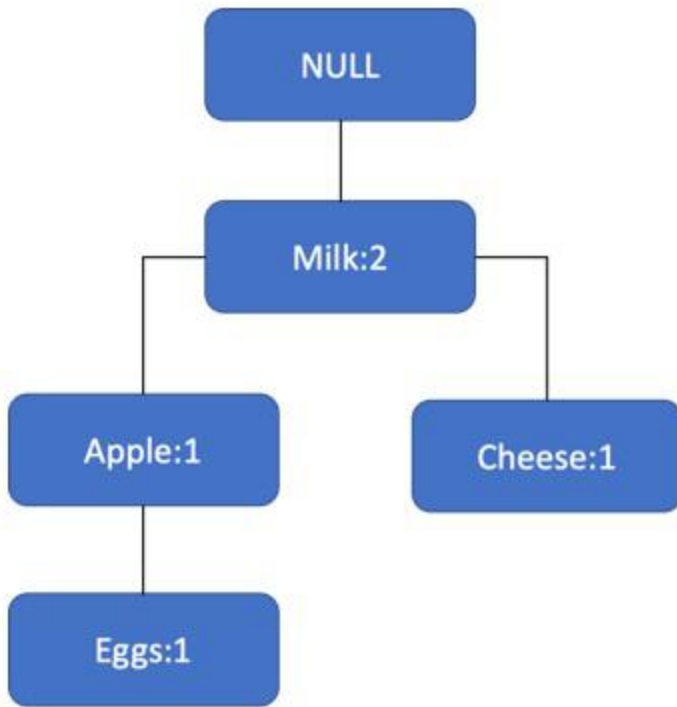
Step 6: It is the turn of T3 now. T3 has Milk and Bread. So, similar to step 5, the count for milk is 3 and it becomes Milk: 3. And similar to step 5, we add another connection from Milk to Bread and call it Bread:1. The updated tree is shown in Figure 4-12.

Figure 4-12 In step 6, T3 is analyzed now. Milk's count increased by one more and becomes 3 while bread is added as a new connection.
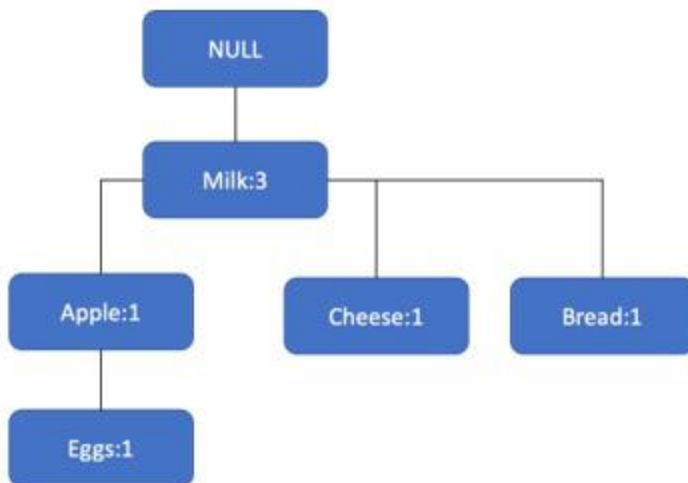
Step 7: In T4, we have Apple, Milk and Cheese. The count for milk becomes 4 now, for apple it is now 2. Then we create a branch from Apple to Cheese calling it Cheese:1. We are showing in Figure 4-13.
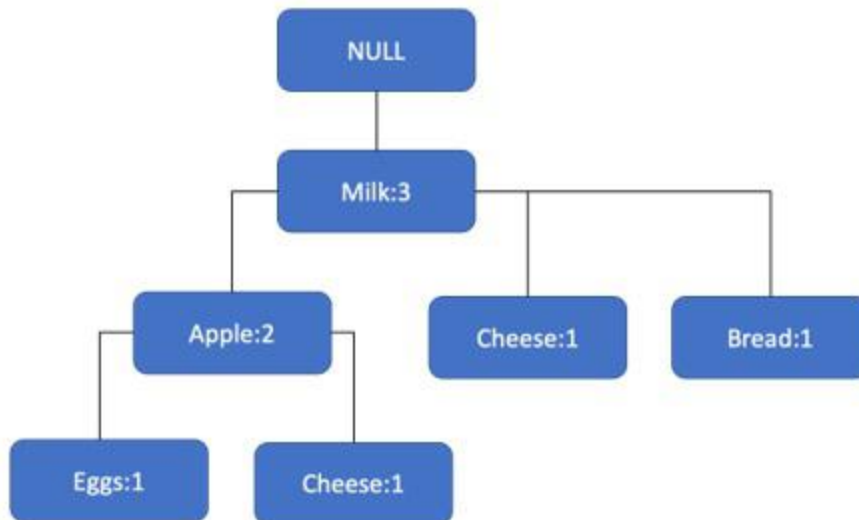
Figure 4-13 In the step 7 of the process, T4 is being analyzed. The count of milk becomes 4, for apple it increases to 2 and a new branch from apple to cheese is added.

Step 8: We can find in T5 that we have Apple and Bread. Both are not directly connected to the root node and have an equal frequency of 6. So, we can take anyone to be connected to the root node. The figure gets updated to Figure 4-14.

Figure 4-14 After analyzing T5, the diagram changes as shown here. We have apple and bread which get added to the tree.
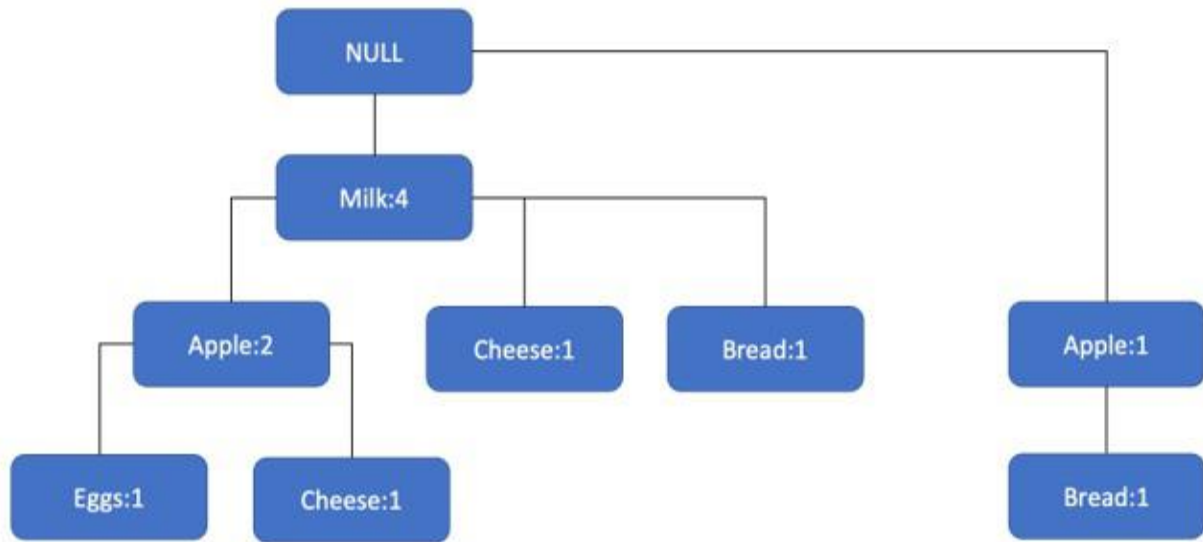
Step 9: And this process continues till we exhaust all the transactions resulting in the final figure as shown in Figure 4-15.

Figure 4-15 The final tree once we have exhausted all the possible combinations. But there are more steps after this. So far, we have created only the tree. Now we need to generate the data set as shown in Table 4-11.
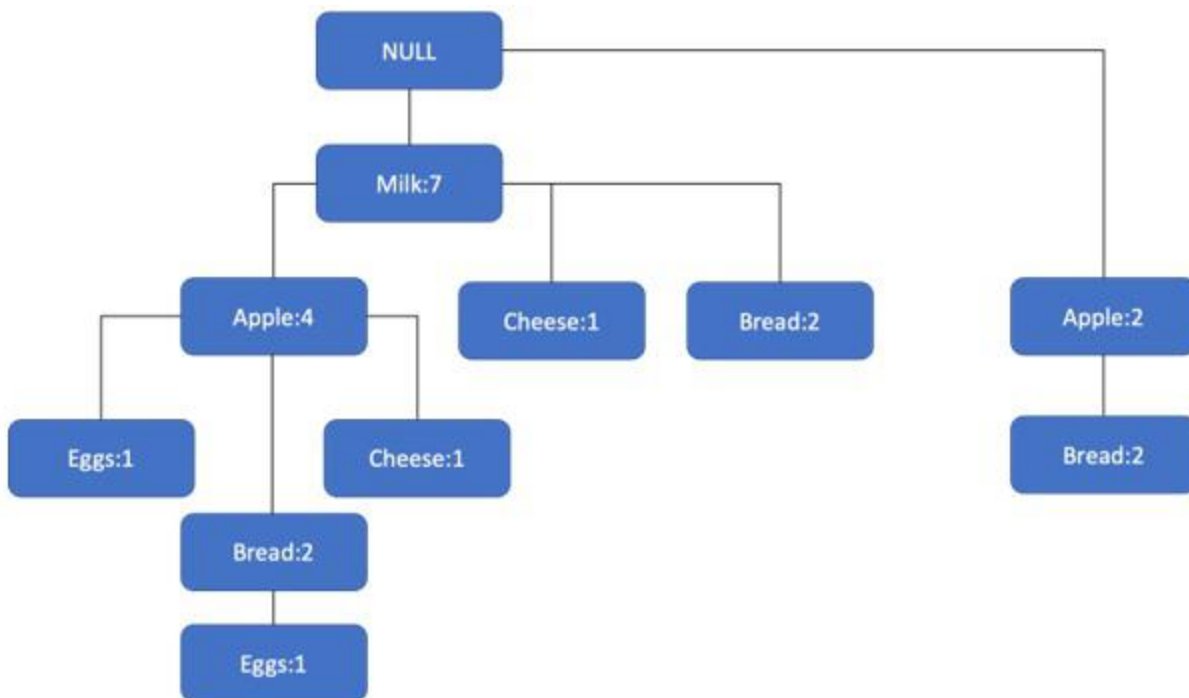
Great job so far! But the process is not over yet. We have just made the

connections between the items in the dataset. Now we need to fill a table which looks like Table 4-11.

Table 4-11 Table we are going to complete for FP algorithm. It is the output we wish to generate.

Items	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern Gener
Cheese			
Bread			
Eggs			
Apple			

You might be wondering that why there are only 4 items listed. Since Milk has directly originated from the root node and there is no other way to reach it, we need not have a separate row for milk.

Step 10: Before starting, we are fixing the minimum support count as 2 for any rule to be acceptable. We are doing it for simplicity sake as the dataset is quite small.

For real-life business problems, you are advised to test with multiple and even much higher values for the support counts otherwise the number of rules generated can be very high.

Let's start with Cheese as the first item. We can reach cheese through {NULL-Milk-Cheese} and {NULL-Milk-Apple-Cheese}. For both the paths, the count of Cheese is 1. Hence, (if we ignore NULL) our conditional pattern base is {Milk-Cheese} or {Milk:1} and {Milk-Apple:Cheese} or {Milk-Apple:1}. The complete conditional pattern base become {{Milk:1},{Milk-

Apple:1}}. This information is added to the second column of Table 4-12.

Table 4-12 Step 10 of the process where we have filled the first cell for cheese. We have filled the first cell for cheese.

Items	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern Ge
Cheese	{{Milk:1},{Milk-Apple:1}}		
Bread			
Eggs			
Apple			

Step 11: Now if we add the two values in conditional pattern base, we would get Milk as 2 and Apple as 1. Since we have set up a threshold for the frequency count of 2, we will ignore the count of Apple. The value for conditional FP tree which is the third column in the table become {Milk:2}. Now we simply add the original item to this which become frequent patten generated or the column 4. The table now is 4-13

Table 4-13 Step 11 of the process where we have finished the details for the item cheese. Similarly, all the other items are going to be analyzed and added to the table.

Items	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern Ge
Cheese	{{Milk:1},{Milk-Apple:1}}	{Milk:2}	{Milk-Cheese:2}

Bread

Eggs

Apple

Step 12: In this similar fashion all the other cells are filled in the table resulting in the final table as Table 4-14.

Table 4-14 Final table after we analyzed all the combinations for the items.

Items	Conditional Pattern Base	Conditional FP Tree
Cheese	{{Milk:1},{Milk-Apple:1}}	{Milk:2}
Bread	{{Milk-Apple:2}, {Milk:2}, {Apple:2}}	{{Milk:4, Apple:2}, {App
Eggs	{{Milk-Apple:1},{Milk-Apple-Bread:1}}	{Milk:2, Apple:2}
Apple	{Milk:4}	{Milk:4}

It is a complex process indeed. But once the steps are clear, it is pretty straightforward one.

As a result of this exercise, we have received the final set of rules as depicted in the final column Frequent Pattern Generated.

Note that none of the rules are similar to each other.

We will be using the final column "Frequent Pattern Generated" as the rules for our dataset.

The Python implementation for FP growth algorithm is quite simple and is easy to compute. In the interest of space, we have uploaded the Jupyter Notebook at the GitHub repo of the chapter.

We will now explore another interesting topic which is sequence rule mining. It is very powerful solution which allows the business to tailor their marketing strategies and product recommendations to the customers.

4.7 Sequence rule mining

Consider this. Netflix would have the transactional database of all the movies ordered by customers over time. If they analyse and finds that 65% of customers who bought a war movie X also bought a romantic comedy Y in the following month, this is a very insightful and actionable understanding. It will allow them to recommend their offerings to the customers, and they can customize their marketing strategy. Isn't it?

So far in the chapter, we have covered three algorithms for association rules. But all the data points were limited to the same dataset and there was no sequencing involved. Sequential pattern mining allows us to analyze a dataset which has a sequence of events happening. By analyzing the data set we can find statistically relevant patterns, which allows us to decipher the entire sequence of events. Obviously, the sequence of events is in a particular order which is a very important property to be considered during sequence rule mining.

Sequence rule mining is different from time-series analysis. To know more about time-series analysis refer to Appendix.

Sequence rule mining is utilized across multiple domains and functions. It can be used in biology to extract information during DNA sequencing or can be used to understand the online search pattern of a user. Sequence rule mining would help us understand what the user is going to search next. During the discussion of association rules, we used the transactions in which

milk, bread, eggs were purchased in the same transaction. Sequence rule mining is an extension to that wherein we analyze consecutive transactions and try to decipher the sequence in the but tears of item sets.

While studying SPADE algorithm, we will study the mathematical concepts which form the base of the algorithm. These concepts are a little tricky to get and might require more than one reading to grasp.

4.7.1 SPADE

We are now exploring the sequence rule mining using Sequential Pattern Discovery using Equivalence classes) or SPADE. It was suggested by Mohammed J. Zaki, the link to the paper is at the end of this chapter.

We understand that we wish to analyze the sequence of events. For example, the customer bought a mobile phone and a charger. After a week bought earphones and after two weeks bought a mobile cover and mobile screen guard. So, in each of the transactions, there are items purchased. And each transaction can be called as an event. Let's understand it in more detail.

Let us assume we have I as the complete list of items for the discussion. I will contain items like i_1, i_2, i_3, i_4, i_5 and so on. So, we can write

I = {i_1, i_2, i_3, i_4, i_5………., i_n} where we have n distinct items in total.

Items can be anything. If we consider the same example of a grocery store, items can be milk, eggs, cheese, bread and so on.

An event will be a collection of items in the same transaction. An event can contain items like (i_1, i_5, i_4, i_8). For example, an event can contain items bought in the same transaction (milk, sugar, cheese, bread). We will denote an event by α.

Next let's understand a sequence. A sequence is nothing but events in an order. In other words, α_1 -> α_2 ->α_3 ->α_4 can be termed as a sequence of event. For example, (milk, cheese) -> (bread, eggs)-> (cheese, bread, sugar)->(milk, bread) is a sequence of transactions. It means that in first transaction

milk, cheese is bought. In the following transaction, bread and eggs were bought and so on.

A sequence with k items is a k-item sequence. For example, sequence {Milk, Bread) -> (Eggs) contain 3 items.

We will now understand SPADE algorithm step-by-step.

Let's say we have the following sequences generated. In the first sequence 1001 of transactions, milk is bought in the very first transaction. In the second one, milk, eggs and bread are bought. They are followed by milk and bread. In fourth one only sugar is purchased. In the fifth and final transaction of sequence 1001, bread and apples are purchased. And this is applicable for all the respective sequences.

Table 4-15 The dataset for sequence mining. In sequence Id 1001, we have multiple events. In the first purchase, milk is bought. Then (milk, eggs, bread) are bought and so on.

Sequence ID Sequence

1001	<(Milk) (Milk, Eggs, Bread) (Milk, Bread) (Sugar)(Bread, Apple
1002	<(Milk, Sugar) (Bread) (Eggs, Bread) (Milk, Cheese)>
1003	<(Cheese, Apple) (Milk, Eggs) (Sugar, Apple) (Bread) (Eggs)>
1004	<(Cheese, Banana)(Milk, Apple)(Bread)(Eggs)(Bread)>

This (Table 4-15) can be converted into a vertical data format as shown in Table 4-16. In this step, we calculate the frequencies for 1-sequence items, which are sequence with only one item. For this only a single database scan is required.

Table 4-16 Vertical format for table 4-15. We have simply got the sequence Id and Item id for each of the item and represented it here.

Sequence ID Element ID Items

Sequence ID	Element ID	Items
1001	1	Milk
1001	2	Milk, Eggs, Bread
1001	3	Milk, Bread
1001	4	Sugar
1001	5	Bread, Apple
1002	1	Milk, Sugar
1002	2	Bread
1002	3	Eggs, Bread
1002	4	Milk, Cheese
1003	1	Cheese, Apple
1003	2	Milk, Eggs

1003	3	Sugar, Apple
1003	4	Bread
1003	5	Eggs
1004	1	Cheese, Banana
1004	2	Milk, Apple
1004	3	Bread
1004	4	Eggs
1004	5	Bread

The Table 4-16 is nothing but a vertical tabular representation of Table 4-15. For example, in sequence Id 1001, at the element ID 1 we have Milk. For sequence ID 1001, at the element ID 2 we have Milk, Eggs, Bread and so on.

For the purpose of explanation, we are considering only two items 0 milk and eggs and the support threshold of 2.

Then, in the next step we will break it down for each of the items. For example, milk appears in sequence Id 1001 and element Id 1, sequence Id 1001 and element Id 2, sequence Id 1001 and element Id 3, sequence Id 1002 and element Id 1 and so on. It results into a table like Table 4-17 where we have shown for Milk and Eggs. It can be applied to all the items in the dataset.

Table 4-17 Respective sequence Ids for milk and eggs. The same can be done across all the items and the sequences.

Milk		Eggs	
Sequence ID	Element ID	Sequence ID	Element ID
1001	1	1001	2
1001	2	1002	3
1001	3	1003	2
1002	1	1003	5
1002	4	1004	5
1003	2		
1004	3		

Now, we wish to count 2-sequences or with 2 item sequences. We can have two sequence – either Milk -> Eggs or Eggs -> Milk. Let's first take Milk-> Eggs.

For Milk -> Eggs we need to have milk in front of eggs. For the same sequence Id, if the element Id of milk is less than element Id of eggs, then it is an eligible sequence. In the example above, for sequence Id 1002, element

Id of milk is 1 while element Id of eggs is 2. So we can add that as the first eligible pair as shown in the first row of Table 4-18. The same is true for sequence Id 1002. In Table 4-17, row 4 we have sequence Id 1002. Element Id of milk is 1 while that of eggs in row 2 is 3. Again element Id of milk is lesser than element Id of eggs, so it becomes the second entry. And the process continues.

Table 4-18 Sequence Milk Eggs can be written down here. The key point is to have the same sequence Id while comparing the respective element Ids of milk and eggs.

Milk Eggs

Sequence ID	Element ID (Milk)	Element ID (Eggs)
1001	1	2
1002	1	3
1003	2	5
1004	3	5

By using the same logic, we can create the table for eggs -> milk which is shown in Table 4-19 below.

Table 4-19 Sequence Eggs Milk can be written down here. The key point is to have the same sequence Id while comparing the respective element Ids of milk and eggs.

Eggs Milk

Sequence ID Element ID (Milk) Element ID (Eggs)

1001   2   3

1002   3   4

This can be done for each of the possible combinations. We will now move to creating 3-item sequences and we will create Milk, Eggs -> Milk. For this purpose, we have to join the two tables.

Table 4-20 Combining both the sequence i.e., milk-> eggs and eggs-> milk to join the tables.

Milk Eggs			Eggs Milk		
Sequenc e ID	Element ID (Milk)	Element ID (Eggs)	Sequenc e ID	Element ID (Eggs)	Element ID (Milk)
1001	1	2	1001	2	3
1002	1	3	1002	3	4
1003	2	5			
1004	3	5			

The logic of joining is matching the sequence Id and the element Id. We have highlighted the matching ones in red and green colour respectively. For sequence Id 1001, the element Id of eggs in the left table matches with element Id of eggs in the right table and that becomes the first entry of Table 4-21. Similarly for sequence Id 1002, element Id 3 matches. This results in the Table 4-21.

Table 4-21 Final table after we analyzed all the combinations for the items.

Milk, Eggs -> Milk

Sequence ID	Element ID (Milk)	Element ID (Eggs)	Element ID (Milk)
1001	1	2	3
1002	1	3	4

This process continues. The algorithm stops when no frequent sequences can be found.

We will now implement SPADE on a dataset using Python. The coding for SPADE is quite simple. We are using pyspade library and hence we have to load the dataset and call the function. It generates the result for us. The support is kept as 0.6 here and then we are printing the results.

```
from pycspade.helpers import spade, print_result
spade_result = spade(filename='SPADE_dataset.txt', support=0.6, p
print_result(spade_result)
```

```
[17]: print_result(spade_result)
         Occurs    Accum    Support     Confid      Lift
      Sequence
            88       88 0.7927928        N/A       N/A
      (10)
            68       68 0.6126126  0.7727273 0.8168831
      (10)->(6)
            67       67 0.6036036  0.7613636 0.7825126
      (10)->(9)
            88       88 0.7927928        N/A       N/A
      (3)
            71       71 0.6396396  0.8068182 0.8292298
      (3)->(9)
           102      560 0.9189189        N/A       N/A
      (4)
            79       79 0.7117117  0.7745098 0.8428489
      (4)->(4)
            77       77 0.6936937  0.7549020 0.7980392
      (4)->(5)
            83       83 0.7477477  0.8137255 0.8602241
      (4)->(6)
            80       80 0.7207207  0.7843137 0.8136339
      (4)->(7)
            77       77 0.6936937  0.7549020 0.7831226
      (4)->(8)
            83       83 0.7477477  0.8137255 0.8363290
      (4)->(9)
```

This concludes our four algorithms which we wish to discuss in this chapter.

We will now move to the case study to give a real-life experience to you.

4.8 Case study for association rules

Association rule mining is quite a helpful and powerful solution. We are going to solve an actual case study using association rules.

Recall that at the start of the chapter we suggested to study the pattern of a grocery store. What is the logic of such arrangements in the store?

Consider this. You are working for a grocery retailer like Walmart or Tesco or Spar or Marks & Spencer's etc. And they have to plan the visual layout of a new store. Obviously, it is imperative that retail stores utilize the space in the store wisely and to the maximum of its capacity. At the same time, it is vital that the movement of the customers is not hindered. Customers should have access to all the items at display and can navigate easily. You might have experienced some stores where we feel choked and bombarded with displays while others are neatly stacked.

How do we solve this problem?

There can be multiple solutions to this problem. Some retailers might wish to group the items based on their categories. They might want to keep all the baking products in one shelf or using any other condition. We are studying the machine learning example here.

Using market basket analysis, we can generate the rules which indicate the respective relationships between various items. We can predict which items are frequently bought together and they can be kept together in the store. For example, if we know that milk and bread are bought together, then bread can be kept near the milk counter. The customer purchasing milk can locate bread easily and continue with their purchase.

But it is not that easy as it sounds. Let us solve this case step-by-step.

1. **Business problem definition** : the very first step is defining the business problem which is clear to us. We wish to discover the

relationships between various items so that the arrangement in the store can be made better. Here *planograms* come into picture. Planogram help the retailer plan the utilization of the space in the store in a wise manner so that the customer can also navigate and access the products easily. It can be considered as a visual layout of the store. An example can be shown as in Figure 4-16.

Figure 4-16 An example of planogram is shown here. Planograms are very useful for visual merchandising.
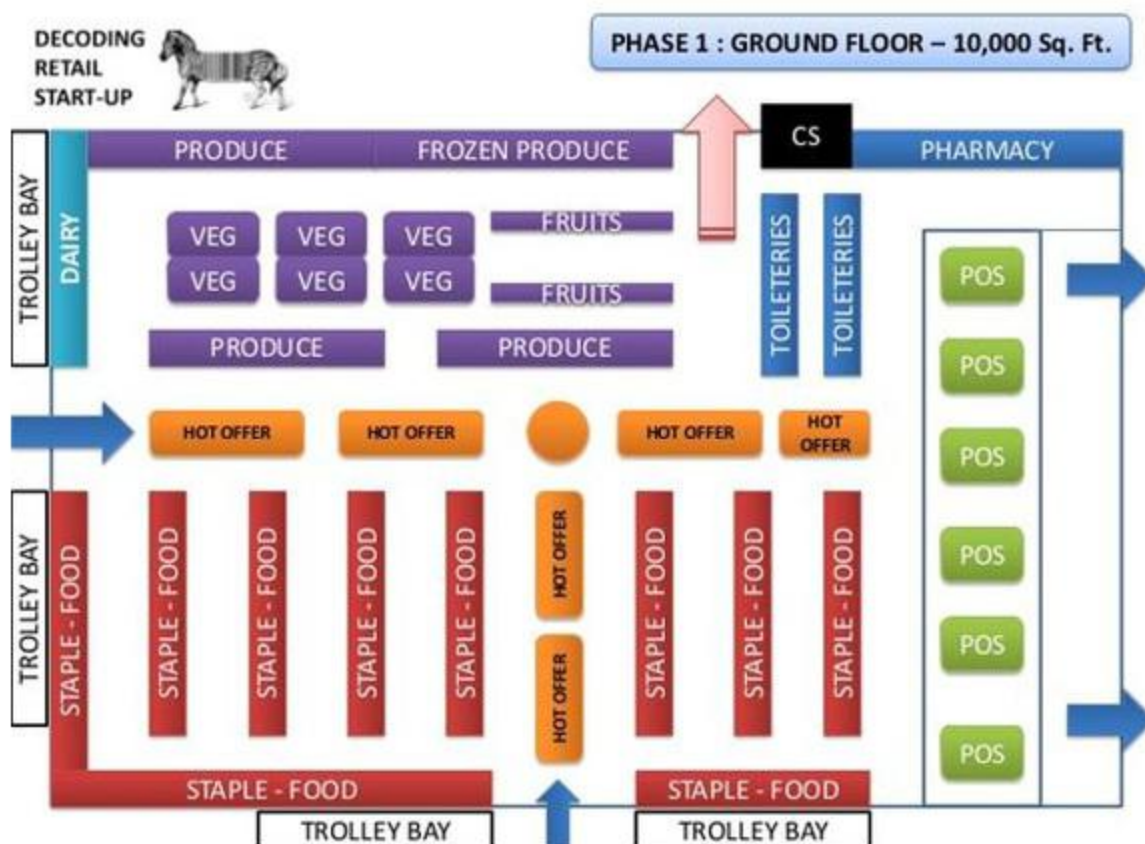
In the figure , we can see that there are specific areas for each and every item categories. Association rules are quite insightful to help generate directions for planogram.

2. **Data discovery** : the next step is the data discovery wherein the historical transactions are scouted and loaded into a database. Typically, a transaction can look like the Table 4-22.

Table 4-22 Example of invoices generated in real-world retail store. It is quite a challenge to convert this data format into the one which can be consumed by the association rule algorithms.

Invoice Number	Date	Items
1001	01-Jun-21	Milk, eggs, cheese, bread
1002	01-Jun-21	Bread, banana, apples, butter
1003	01-Jun-21	Butter, carrots, cheese, eggs, bread, milk, bananas
1004	01-Jun-21	Milk
1005	01-Jun-21	Bread

3. **Data preparation**: this step perhaps is the most difficult step. As you would have realised that association rules model creation is a vey simple task. We have libraries which can do the heavy lifting for us. But the data set expected by them is in a particular format. This is a tedious task, quite time consuming and requires a lot of data pre-processing skills. There are a few considerations you have to keep in mind while preparing the data set, which are:
4. Sometimes we get **NULL** or **blank values** during the data preparation phase. Missing values in the data sets can lead to problems while computing. In other machine learning solution, we would advise to treat the missing values. In the case of association rules, we would suggest to ignore the respective transactions and do not consider it in the final dataset.
5. Many times, we get **junk values** in the data. Special characters like !@%^&*()_ are found in the datasets. It can be attributed to incorrect entries in the system. Hence, data cleaning is required.

6. We are covering the data pre-processing step in great detail in the appendix of the book, wherein we deal with NULL values and junk values.
7. Converting Table into a format which can be understood and consumed by the association rule learning algorithms is an imperative but arduous step. Go through the concept of SQL pivoting to understand the concept better. Else, you might need someone (a data engineer) to create the dataset for you.
8. **Model preparation :** perhaps the easiest of the steps is modelling. We have already solved Python solutions for different algorithms. So, you should be quite comfortable with it.
9. **Model interpretation** : creating the model might be easy but interpretation of the rules is not. Most of the time, you can rules like:
10. #NA -> (Milk, Cheese) – such a rule is obviously non-usable and does ot make any sense. It indicated that the data preparation was not correct and some junk values are still present in the dataset.
11. (Some items) -> (Packaging material) – perhaps the most obvious rule but again not usable. This rule indicates that whenever shopping is done, packaging material is also purchased, quite obvious right?
12. (Potatoes, Tomatoes) -> (Onions) : this kind of rule might look correct but it a common sense knowledge which the retailer would already know. Obviously, most of the customer who are buying vegetables will buy potatoes, tomatoes and onion together. Such rules, might not add much value to the business.
13. Threshold for support, confidence and lift allows to filter out the most important rules. We can sort the rules in the descending order of the lift and then remove the most obvious ones.
14. **Business subject-matter expert** : it is of vital importance that business stakeholders and subject matter experts are involved at each and every step. In this case study, operations team, visual merchandising team, product teams and marketing teams are the key players which should be closely aligned at each and every step.
15. Once the rules are generated and accepted, then we can use them to improve the planogram for the retail space. The retailer can use them to improve the marketing strategy and improve the product promotions. For example, if a rule like (A, B) -> (C) is accepted, the retailer might wish to create a bundle of the products and sell them as a single entity. It

will increase the average items purchased in the same transaction for the business.

16. This case study can be extended to any other domain or business function. For example, the same steps can be used if we wish to examine user's movement across web pages. Web developers can analyse the historical clicks and usages of the customers on their websites. By identifying the patterns, they can find out what user tends to click and which features will maximise their engagements. Medical practitioners can use association rules to better diagnose patients. The doctors can compare the probability of the symptoms in relationships with other symptoms and provide more accurate diagnosis.

We will now examine limitations of these algorithms and other solutions which are available for association rules and sequence rules.

4.9 Limitations and Summary

There are some assumptions and limitations in the association rules and sequence rules we have studied.

1. The respective significance of an item is ignored while we generate the rules. For example, if a customer purchased 5 cans of milk and one kg of apples in a transaction, it is treated similar to an invoice in which one can of milk and five kg of apples are purchased. Hence, we have to bear in mind that the respective *weight* of an item in not being considered.
2. The cost of an item indicated the perceived value of a product. Some products which are costly are more important and hence, if they are purchased by the customer more revenue can be generated. While analysing the invoices, we ignore the cost associated with an item.
3. While analysing the sequence, we have not considered the respective time periods between the two transactions. For example, if between T1 and T2, there were 10 days while between T2 and T3 there were 40 days – both are considered as same.
4. In all the analysis, we have considered different categories as same. Perishable items and non-perishable items are treated in a similar fashion. For example, milk with a shelf life of 2-3 days is treated similar to washing powder.

5. Many times we receive non-interesting rules after analysis. These results are from common sense (Potatoes, Tomatoes) -> (Onion). Such rules are not of much use. We face such an issue a lot of the time.
6. While non-interesting rules are a challenge, huge number of discovered rules are again one of the problems. We get hundreds of rules and it becomes difficult to understand and analyse each one of them. Here, the thresholding becomes handy.
7. The time and memory requirements for computations are huge. The algorithms require to scan the data sets many times and hence it is quite a time consuming exercise.
8. The rules generated are dependent on the data set which has been used for analysis. For example, if we analyse the data set generated during summers only, we cannot use the rules for winters as consumers preference change between different weathers. Moreover, we have to refresh the algorithms over time since with the passage of time, the macro and micro economic factors change and hence, the algorithms should be refreshed too.

There are some other algorithms which are also of interest. For association rules, we can have multi-relation association rules, k-optimal pattern discovery, approximate frequent data set, generalized association rules, high order pattern discovery etc. For sequence mining, we have Generalized Sequence Pattern, FreeSpan, PrefixSpan, Mining associated patterns etc. These algorithms are quite interesting and can be studied for knowledge enhancement.

Association rules and sequence mining are quite interesting topics. Various business domains and functions are increasingly using association rules to understand the pattern of events. These insights allow the teams to take sound and scientific decisions to improve the customer experience and overall engagement. This chapter is the first chapter in the second part of the book. We have explored association rules and sequence mining in this chapter. These are studied using Apriori, FP and ECLAT algorithms and for sequence mining we used SPADE.

In the next chapter, we are studying advanced clustering algorithms. So stay tuned!

You can now progress to question section.

Practical next steps and suggested readings

1. Go through these research papers for association rules algorithm
2. Fast discovery of association rules (http://www.cs.bme.hu/~marti/adatbanya/apriori_hashtree.pdf)
3. Fast algorithms for Mining Association Rules (https://rakesh.agrawal-family.com/papers/vldb94apriori.pdf)
4. Efficient analysis of Pattern and Association Rule Mining Approaches (https://arxiv.org/pdf/1402.2892.pdf)
5. A review of association rule mining techniques with respect to their privacy preserving capabilities (https://www.ripublication.com/ijaer17/ijaerv12n24_216.pdf)
6. For sequence mining, go through these research papers:
7. SPADE: An efficient algorithm for mining frequent sequences (https://link.springer.com/content/pdf/10.1023/A:1007652502315.pdf)
8. Sequential mining: patterns and algorithm analysis (https://arxiv.org/pdf/1311.0350.pdf)
9. Sequential pattern mining algorithm based on interestingness (https://ieeexplore.ieee.org/document/8567170)
10. A new approach for problem of Sequential Pattern Mining (https://link.springer.com/chapter/10.1007/978-3-642-34630-9_6)

5 Clustering (advanced)

"Out of complexity, find simplicity– Einstein"

Sometimes life is very simple, and sometimes we experience quite complex situations. We sail through both the situations and change our approach as per the situation.

In the Part one of the book we covered easier and simpler topics. It made you ready for the journey ahead. We are currently in Part two which is slightly more complex than Part one. Part three is more advanced than the first two parts. So, the level of difficulty will increase slightly with each and every chapter along with the expectations.

We studied clustering algorithms in part one of the book. We understand that clustering is an unsupervised learning technique where we wish to group the data points by discovering interesting patterns in the datasets. We went through the meaning of clustering solutions, different categories of clustering algorithm and a case study at the end. In that chapter, we explored kmeans clustering, hierarchical clustering and DBSCAN clustering in depth. We went through the mathematical background, process, Python implementation and pros and cons of each. Before starting this chapter, it is advisable to refresh chapter two.

Many times you might encounter datasets which do not conform to a simple shape and form. Moreover, we have to find the best fit before making a choice of the final algorithm we wish to implement. Here, we might need help of more complex clustering algorithms; the topic for the chapter. In this chapter, we are going to again study three such complex clustering algorithms – spectral clustering, Gaussian Mixture Models (GMM) clustering and fuzzy clustering. As always, Python implementation will follow the mathematical and theoretical concepts. This chapter is slightly heavy on the mathematical concepts. There is no need for you to be a PhD. in mathematics, but it is sometime important to understand how the algorithms work in the background. At the same time, you will be surprised to find that

Python implementation of such algorithms is not tedious. This chapter is not having any case study.

In this fifth chapter of the book, we are going to cover the following topics:

1. Spectral clustering
2. Fuzzy clustering
3. Gaussian Mixture Models (GMM) clustering
4. Summary

Welcome to the fifth chapter and all the very best!

5.1 Technical toolkit

We will continue to use the same version of Python and Jupyter notebook as we have used so far. The codes and datasets used in this chapter have been checked-in at GitHub (https://github.com/vverdhan/UnsupervisedLearningWithPython/tree/main/Ch

We are going to use the regular Python libraries we have used so far – numpy, pandas, sklearn, seaborn, matplotlib etc. You would need to install a few Python libraries in this chapter which are – skfuzzy and network . Using libraries, we can implement the algorithms very quickly. Otherwise, coding these algorithms is quite a time-consuming and painstaking task.

Let's get started with a refresh of clustering!

5.2 Clustering

Recall from chapter 2, clustering is used to group similar objects or data points. It is an unsupervised learning technique where we intend to find natural grouping in the data as shown in Figure 5-1.

Figure 5-1 Clustering of objected result into natural grouping.
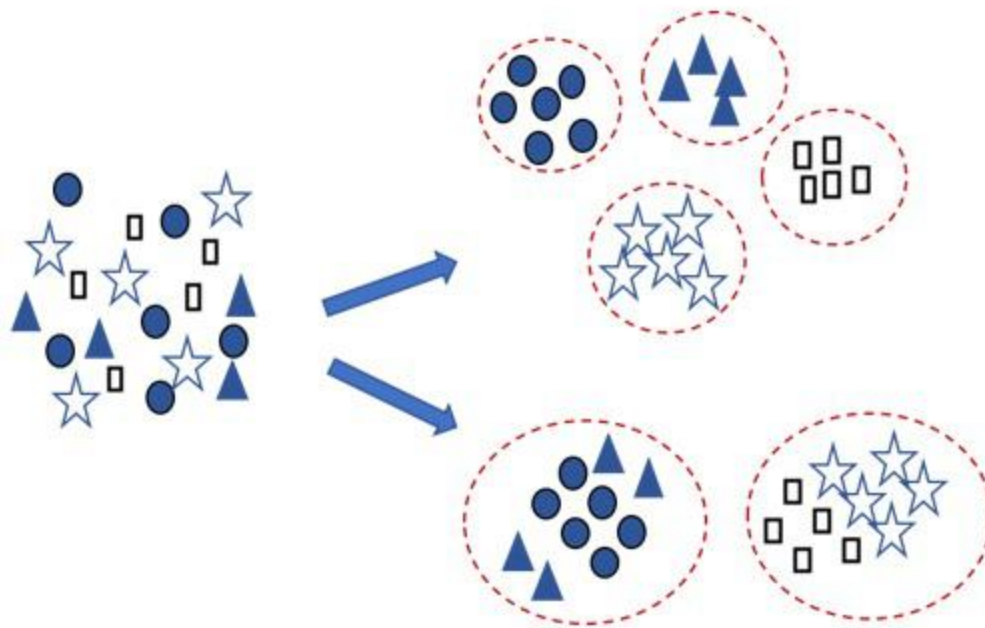
Here we can observe that on the left side we have ungrouped data and on the right side the data points have been grouped into logical groups. We can also observe that there can be two methodologies to do the grouping or *clustering*, and both result into different clusters. Clustering as a technique is quite heavily used in business solutions like customer segmentation, market segmentation etc.

We have understood kmeans, hierarchical and DBSCAN clustering in chapter 2. We also covered various distance measurement techniques and indicators to measure the performance of clustering algorithms. You are advised to revisit the concepts.

In this chapter, we are focussing on advanced clustering methods. We will now start with Spectral clustering in the next section.

5.3 Spectral Clustering

Spectral clustering is one of the unique clustering algorithms. There are some good quality research is done in this field. Revered researchers like Prof. Andrew Yang, Prof. Michael Jordan, Prof. Yair Weiss, Prof. Jianbo Shi, Prof. Jitendra Malik to name a few. We are quoting some of the papers in the last

section of the chapter.

Let's define spectral clustering first. Spectral clustering works on the affinity and not the absolute location of the data points for clustering. And hence, wherever the data is in complicated shapes, spectral clustering is the answer. We have shown a few examples in the Figure 5-2 where spectral clustering can provide a logical solution.

Figure 5-2 Examples of various complex data shapes which can be clustered using Spectral Clustering.
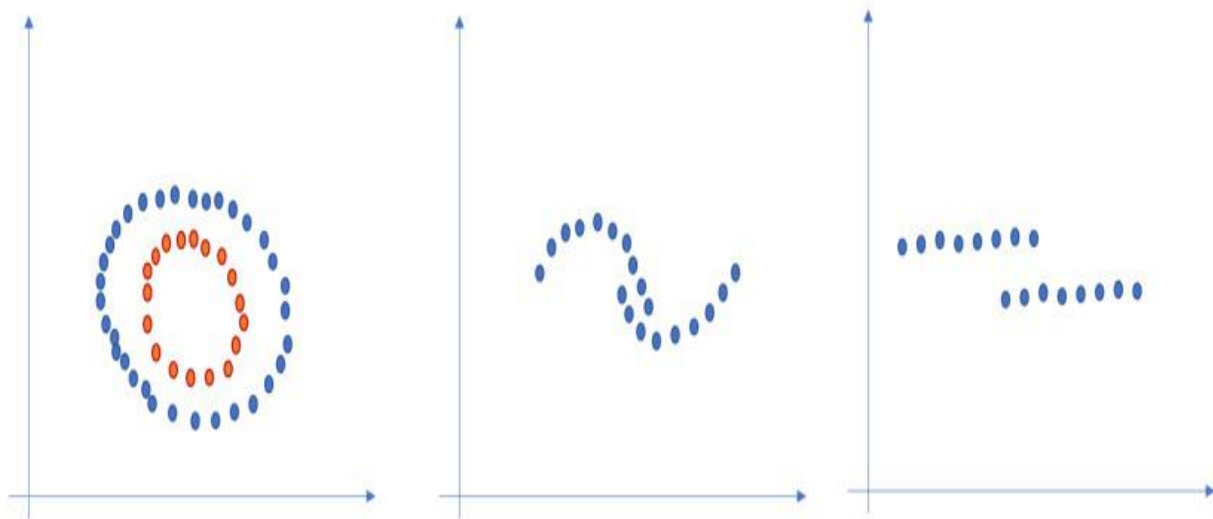

For Figure 5-2, we could have used other algorithms like k-means clustering too. But they might not be able to do justice to such complicated shapes of the data. Formally put, algorithms like kmeans clustering utilize compactness of the data points. In other words, the closeness of the points to each other and compactness towards the cluster center, drive the clustering in kmeans. On the other hand, in Spectral clustering *connectivity* is the driving logic. In connectivity, either the data points are immediately close to one another or are connected in some way. The examples of such connectivity-based clustering have been depicted in Figure 5-2.

Look at the Figure 5-3(i) where the data points are in this doughnut pattern. There can be data points which can follow this doughnut pattern. It is a complex pattern and we have to cluster these data points. Imagine that by using a clustering method, the red circles are made a part of the same cluster,

which is shown in Figure 5-3(ii). After all, they are close to each other. But if we look closely, the points are in a circle, are in a pattern and hence the actual cluster should be as shown in Figure 5-3(iii).

Figure 5-3 (i) We can have such a complex representation of data points which need to be clustered. Observe the doughnut shape (ii) A very simple explanation can result in red dots being considered as a part of the same cluster but clearly, they are not part of the same cluster (iii) We have two circles over here. The points in the inner circle belong to one cluster whereas the outer points belong to another cluster
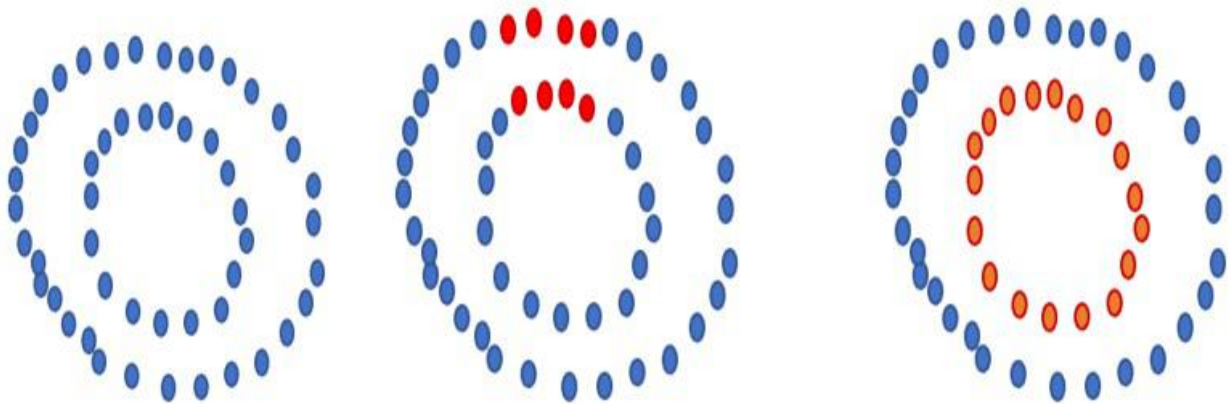
The example shown in Figure 5-3 is to depict the advantages with Spectral clustering.

Like we said earlier, spectral clustering utilizes connectivity approach in clustering. In spectral clustering, data points which are immediately next to each other are identified in a graph. These data points are sometimes referred to as *node*. These data points or nodes are then mapped to a low-dimensional space. A low dimensional space is nothing but having During this process, spectral clustering uses eigenvalues, affinity matrix, Laplacian matrix and degree matrix derived from the data set. The low-dimensional space can then be segregated into clusters.

Spectral clustering utilizes connectivity approach for clustering wherein It relies on graph theory wherein we identify clusters of nodes based on the edges connecting them.

We will study the process in detail. But before examining the process, there are a few important mathematical concepts which form the foundation of

spectral clustering which we will cover now.

5.3.1 Building blocks of Spectral Clustering

We know that the goal of clustering is to group data points which are similar into one cluster, while the data points which are not similar into another one. There are a few mathematical concepts we should be aware. We will start the concept of similarity graphs, which is quite an innate representation for data points.

Similarity graphs

A graph is one of the easiest and intuitive method to represent data points. In the Figure 5-4(i), we are showing an example of a graph which is nothing but a connection between data points represented by the edge. Now two data points are connected if the similarity between them is positive or it is above a certain threshold which is shown in Figure 5-4(ii). Instead of absolute values for the similarity, we can use weights for the similarity. So, in Figure 5-4(ii), as point 1 and 2 are similar as compared to point 1 and 3, the connection between point 1 and 2 has a higher weight than point 1 and 3.

Figure 5-4(i) A graph is a simple representation of data points. The points or nodes are connected with each other if they are very similar (ii) The weight is higher if the similarity between data points is high else for dissimilar data points, the weight is less.
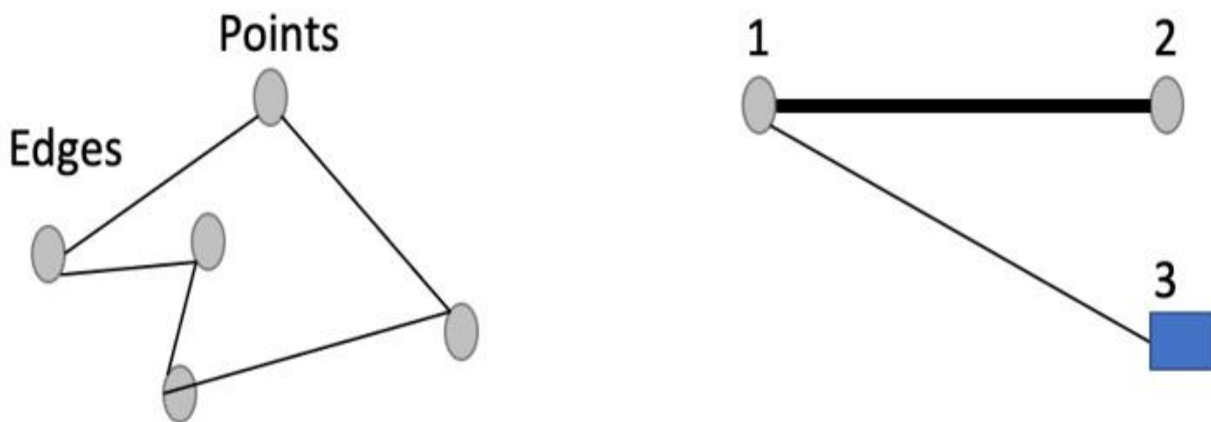
So, we can conclude that – using Similarity Graphs we wish to cluster the

data points such that

- the edges of the data points have higher value of weight and hence are similar to each other and so are in the same cluster.
- the edges of the data points have lower values of weight and hence are not similar to each other and so they are in different clusters.

Apart from similarity graphs, we should also know the concept of **Eigen values and Eigen vectors** which we have covered in detail in the previous chapter. You are advised to refresh it. We will now move to the concept of Adjacency matrix.

Adjacency Matrix

Have a close look at Figure 5-5. We can see those various points from 1 to 5 are connected with each other. And then we are representing the connection in a matrix. That matrix is called *Adjacency Matrix*.

Formally put, in adjacency matrix, the rows and columns are the respective nodes. The values inside the matrix represent the connection – if the value is 0 that means there is no connection and if the value is 1, it means there is a connection.

Figure 5-5 Adjacency matrix represents the connection between various nodes, if the value is 1 that means the corresponding nodes in the row and column are connected. If the value is 0, it means they are not connected. For example, there is a connection between node 1 and node 5 hence the value is 1 while there is no connection between node 1 and node 4 hence the corresponding value is 0.
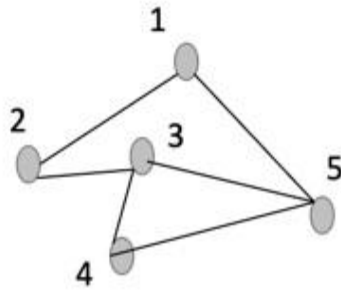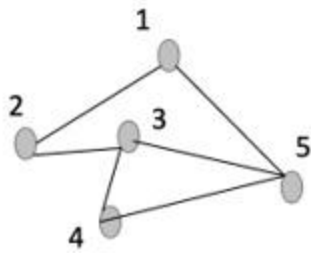
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	0	0
3	0	1	0	1	1
4	0	0	1	0	1
5	1	0	1	1	0

So for adjacency matrix, we are only concerned if there is a connection between two data points. If we extend the concept of adjacency matrix, we get degree matrix which is our next concept.

Degree Matrix

Formally put, a degree matrix is a diagonal matrix, where the degree of a node along the diagonal is the number of edges connected to it. If we use the same example as above, we can have the degree matrix as shown in Figure 5-6. Node 3 and 5 have three connections each and they are getting 3 as the values along the diagonal while the other nodes have only two connections each, so they have 2 as the value along the diagonal.

Figure 5-6 While adjacency matrix represents the connection between various nodes, degree matrix is for the number of connections to each node. For example, node 5 has three connections and hence has 3 in front of it while node 1 has only two connections so it has 2.

	1	2	3	4	5
1	2	0	0	0	0
2	0	2	0	0	0
3	0	0	3	0	0
4	0	0	0	2	0
5	0	0	0	0	3

You might be wondering why do we use matrix? Matrix provide an elegant representation of the data and can clearly depict the relationships between two points.

Now we have covered both adjacency matrix and degree matrix, we can move to Laplacian matrix.

Laplacian Matrix

There are quite a few variants of Laplacian matrix, but if we take the simplest form of Laplacian matrix, it is nothing but a subtraction of adjacency matrix from the degree matrix. In other words, L = D – A. We can show it as Figure 5-7.

Figure 5-7 Laplacian matrix is quite simple to understand. To get a Laplacian matrix, we can simply subtract an adjacency matrix from the degree matrix as shown in the example above. Here, D represents the degree matrix, A is the adjacency matrix and L is the Laplacian matrix.

D	1	2	3	4	5
1	2	0	0	0	0
2	0	2	0	0	0
3	0	0	3	0	0
4	0	0	0	2	0
5	0	0	0	0	3

−

A	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	0	0
3	0	1	0	1	1
4	0	0	1	0	1
5	1	0	1	1	0

=

L	1	2	3	4	5
1	2	-1	0	0	-1
2	-1	2	-1	0	0
3	0	-1	3	-1	-1
4	0	0	-1	2	-1
5	-1	0	-1	-1	3

Laplacian matrix is quite an important one and we use the eigen values of L to develop spectral clustering. Once we get the eigen values and eigen vectors, we can define two other values – spectral gap and Fielder value. The very first non-zero eigen value is the *Spectral Gap* which defines the density of the graph. The *Fielder value* is the second eigen value which provides us an approximation of the minimum cut required to separate the graph into two components. The corresponding vector for Fielder value is called the *Fielder vector*.

Fielder vector has both negative and positive components and their resultant sum is zero.

We will use this concept once we study the process of Spectral clustering in detail in the next section. We will now cover one more concept of Affinity matrix before moving to the process of Spectral clustering.

Affinity Matrix

In the adjacency matrix, if we replace the number of connections with the similarity of the weights, we will get affinity matrix. If the points are completely dissimilar, the affinity will be 0 else if they are completely similar the affinity will be 1. The values in the matrix represent different levels of similarity between data points.

🎯 POP QUIZ – answer these question to check your understanding..
Answers at the end of the book

1. The degree matrix is created by counting the number of connections. True or False.

2. Laplacian is a transpose of the division of degree and adjacency matrix. True or False.

3. Write a matrix on a paper and then derive its adjacency and degree matrix.

We have now covered all the building blocks for Spectral clustering. We can now move to the process of Spectral clustering.

5.3.2 Process of Spectral Clustering

Now we have covered all the building blocks for Spectral clustering. At a high level, the various steps can be noted as:

1. We get the dataset and calculate its degree matrix and Adjacency matrix.
2. Using them, we get the Laplacian matrix.
3. Then we calculate the first k eigen vectors of the Laplacian matrix. The k eigenvectors are nothing but the ones which correspond to the k smallest eigen values.
4. The matrix such formed is used to cluster the data points in k-dimensional space.

We will now cover the process of Spectral clustering using an example as shown in Figure 5-8. These are the steps which are generally not followed in real-world implementation as we have packages and libraries to achieve it. These steps are covered here to give you the idea of how the algorithm can be developed from scratch. For the Python implementation, we will use the libraries and packages only. Though it is possible to develop an implementation from scratch, it is not time efficient to reinvent the wheel.

Figure 5-8 Consider the example shown where we have some data points and they are connected with each other. We will perform Spectral clustering on this data.
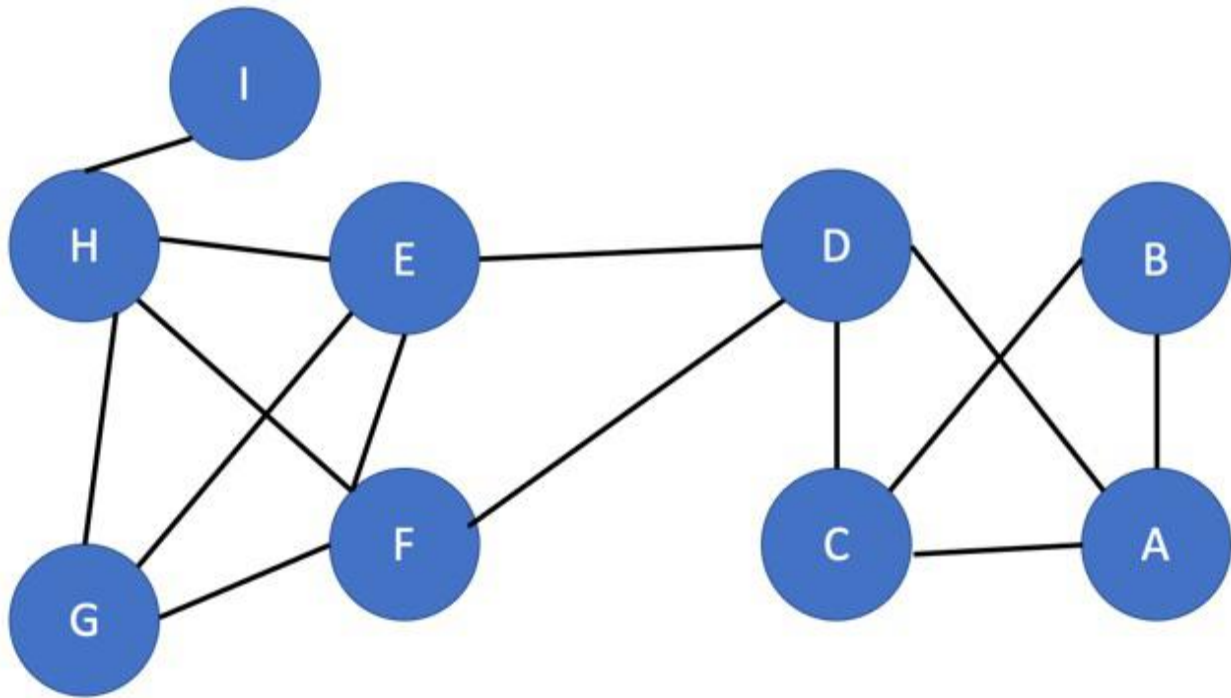
Now when we wish to perform the spectral clustering on this data.

1. We will leave it upto you to create the adjacency matrix and degree matrix.
2. The next step is creating the Laplacian matrix. We are sharing the output Laplacian matrix in Figure 5-9.

Figure 5-9 Laplacian matrix of the data is shown here. You are advised to create the degree and adjacency matrix and check the output.

|	A	B	C	D	E	F	G	H	I
A	3	-1	-1	-1	0	0	0	0	0
B	-1	2	-1	0	0	0	0	0	0
C	-1	-1	3	-1	0	0	0	0	0
D	-1	0	-1	4	-1	-1	0	0	0
E	0	0	0	-1	4	-1	-1	-1	0
F	0	0	0	-1	-1	4	-1	-1	0
G	0	0	0	0	-1	-1	4	-1	-1
H	0	0	0	0	-1	-1	-1	3	0
I	0	0	0	0	0	0	-1	0	-1

3. Now, the Fielder vector is shown in Figure 5-10 for the above Laplacian matrix. We create the Fielder vector as described in the last section. Observe how the sum of the matrix is zero.

Figure 5-10 Fielder vector is the output for the Laplacian matrix, observe that the sum is zero here.

0.33	-0.38
0.33	-0.48
0.33	-0.38
0.33	-0.12
0.33	0.16
0.33	0.30
0.33	0.24
0.33	0.51
0.33	0.16

4. We can see that there are a few positive values and a few negative values, based on which we can create two distinct clusters. This is a very simple example to illustrate the process of Spectral Clustering.

Figure 5-11 The two clusters are identified. This is a very simple example to illustrate the process of Spectral Clustering.
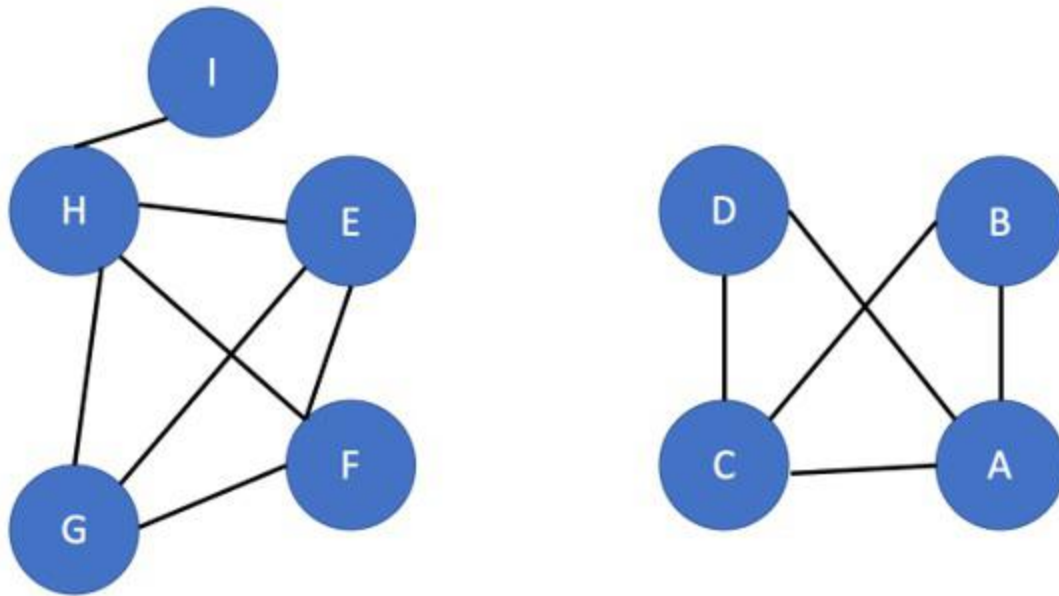
The above process is a very simple representation of Spectral Clustering. Spectral clustering is useful for image segmentation, speech analysis, text analytics, entity resolution etc. It is quite easy and intuitive method and does not make any assumption about the shape of the data. Methods like kmeans assume that the points are in a spherical form around the center of the cluster whereas there is no such strong assumption in Spectral clustering.

Another significant difference is that in spectral clustering the data points need not have convex boundaries as compared to other methods where compactness drives clustering. Spectral clustering is sometimes slow since eigne values, Laplacians etc. have to be calculated. With a large dataset the complexity increases and hence Spectral clustering can become slow, but it a fast method when we have a sparse dataset.

We will now proceed to Python implementation of Spectral Clustering algorithm.

## 5.2.1   Python implementation of Spectral Clustering

We have covered so far, the theoretical details of Spectral Clustering, it is time to get into the code. For this, we will curate a dataset and run k-means algorithm. And then Spectral Clustering to compare the results.

Step 1: Import all the necessary libraries first. These libraries are standard libraries except a few which we will cover. `sklearn` is one of the most famous and sought-after libraries and from `sklearn` we are importing `SpectralClustering`, `make_blobs` and `make_circles`.

```
from sklearn.cluster import SpectralClustering
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from numpy import random
import numpy as np
from sklearn.cluster import SpectralClustering, KMeans
from sklearn.metrics import pairwise_distances
from matplotlib import pyplot as plt
import networkx as nx
import seaborn as sns
```

Step 2: We will now curate a dataset. We are using `make_circles` method. Here we are taking 2000 samples and representing them in a circle. The output is shown below.
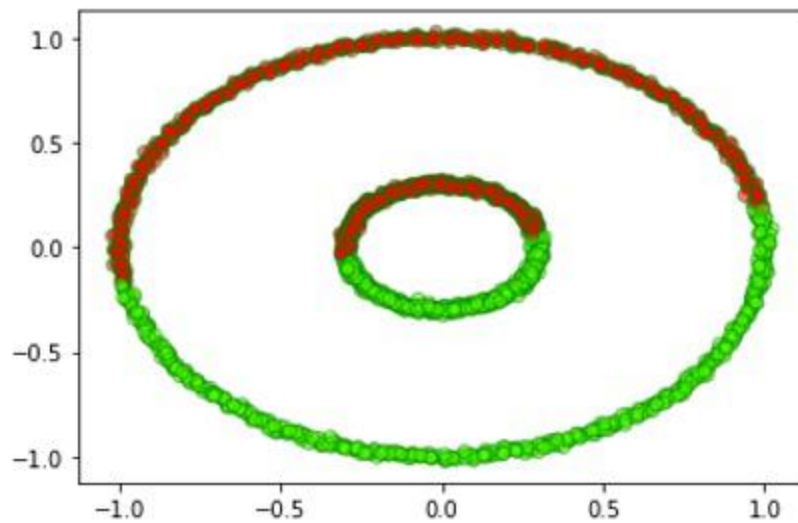
```
data, clusters = make_circles(n_samples=2000, noise=.01, factor=.
plt.scatter(data[:,0], data[:,1])
```
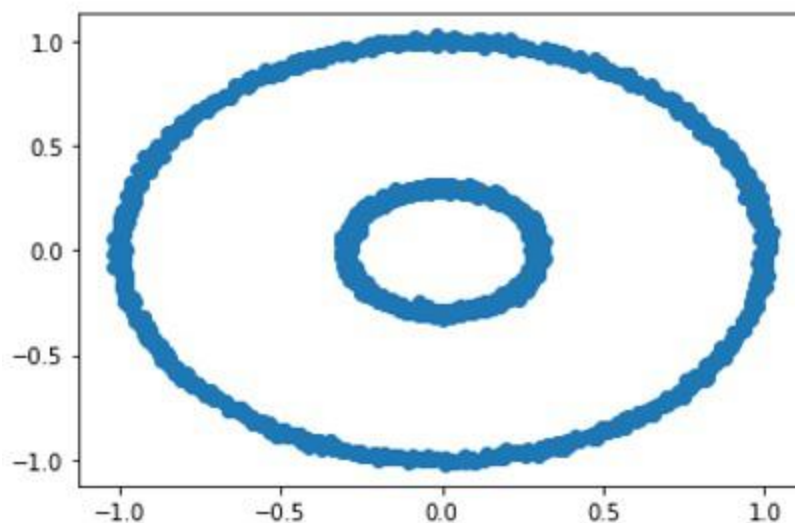

Step 3: We will now test this dataset with kmeans clustering. The two colors are showing two different clusters which are overlapping with each other.

```
kmeans = KMeans(init='k-means++', n_clusters=2)
```

```
km_clustering = kmeans.fit(data)
plt.scatter(data[:,0], data[:,1], c=km_clustering.labels_, cmap='
```


Step 4: We will now run the same data with Spectral Clustering and we find that the two clusters are being handled separately here.

```
spectral = SpectralClustering(n_clusters=2, affinity='nearest_nei
sc_clustering = spectral.fit(data)
plt.scatter(data[:,0], data[:,1], c=sc_clustering.labels_, cmap='
```
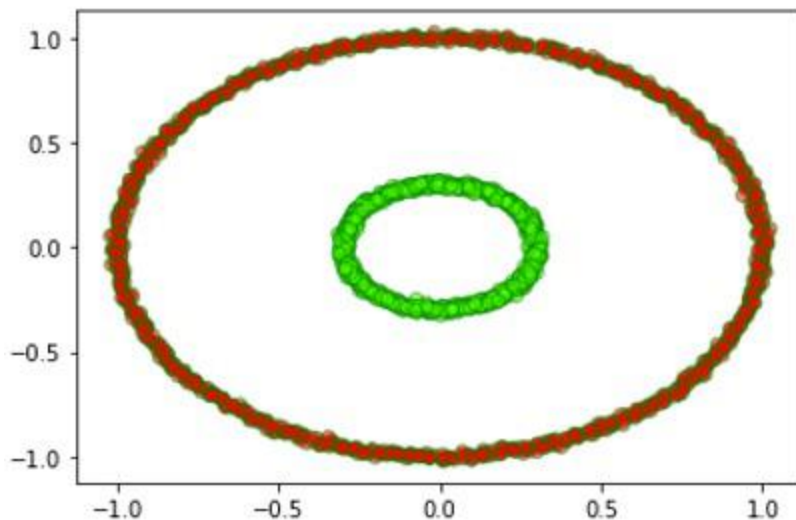

We can observe here that the same dataset is handled differently by the two algorithms. Spectral clustering is handling the dataset better as the circles which are separate are depicted separately.

Step 5: You are advised to simulate various cases by changing the values in the dataset and run the algorithms. Observe the different outputs for comparison.

With this we have finished the first algorithm in this chapter. We will now move to Fuzzy Clustering in the next section.

5.3 Fuzzy Clustering

So far, we have covered quite a few clustering algorithms. Did you wonder that why a data point should belong to only one cluster? Why can't a data point belong to more than one clusters? Have a look at Figure 5-12.

Figure 5-12 The figure of the left represents all the data points. The red points can belong to more than one clusters. In fact, we can allocate more than one cluster to each and every point. A probability score can be given for a point to belong to a particular cluster.
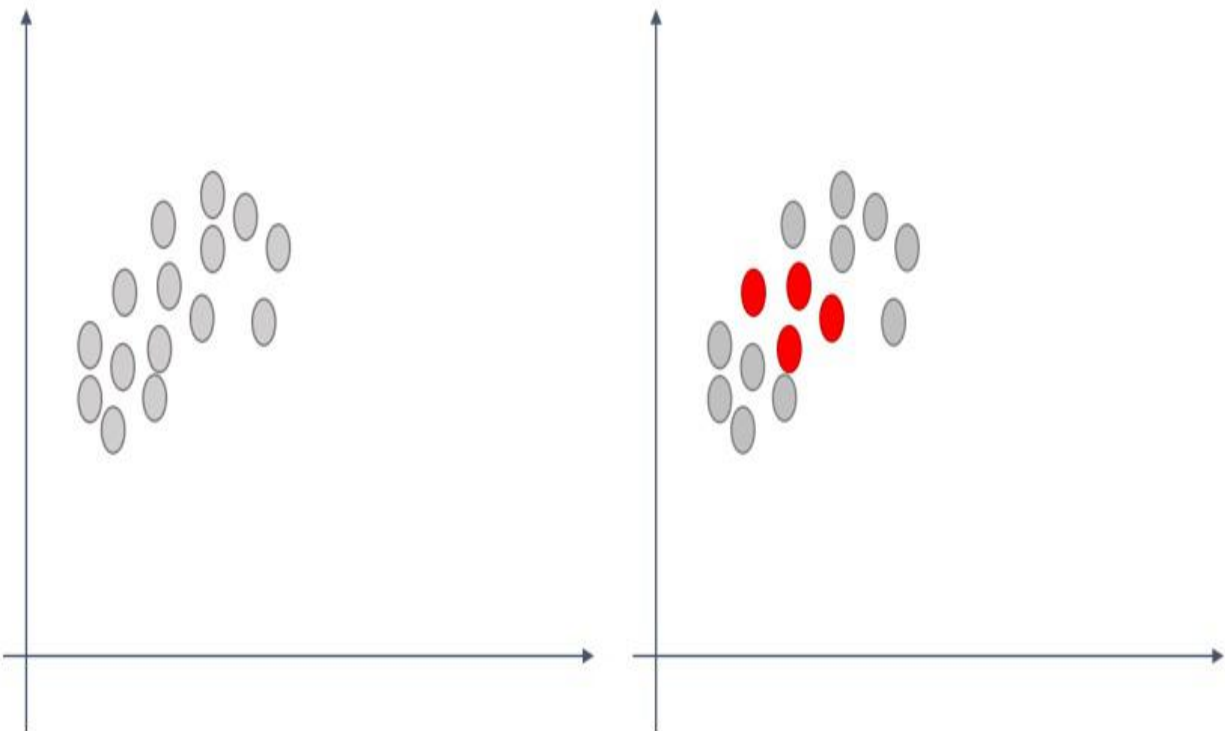
We know that clustering is used to group items in cohesive groups based on the similarities between them. The items which are similar are in one cluster, whereas the items which are dissimilar are in different clusters. The idea of

clustering is to ensure the items in same cluster should be as much similar to each other. When the items can be only in one cluster, it is called as *hard clustering.* K-means clustering is a classic example of hard clustering. But if we reflect back on the Figure 5-12, we can observe that an item can belong to more than one clusters. It is also called *soft clustering.*

It is computationally cheaper to create fuzzy boundaries than create hard clusters.

In fuzzy clustering, an item can be assigned to more than one cluster. The items which are closer to the center of a cluster, might be the part of the cluster to a higher degree than the items which are near the cluster's edge. It is referred as *membership*. It employs least-square solutions to the most optimal location of an item. This optimal location might be the probability space between the two or more clusters. We will examine this concept in detail when we study the process of fuzzy clustering in detail and now, we will move to types of fuzzy clustering algorithms.

5.3.3 Types of Fuzzy Clustering

Fuzzy clustering can be further divided between classical fuzzy algorithms and shape-based fuzzy algorithms which we are showing by means of a diagram Figure 5-13.

Figure 5-13 Fuzzy algorithms can be divided into Classical Fuzzy algorithm and Shape-based fuzzy algorithm.
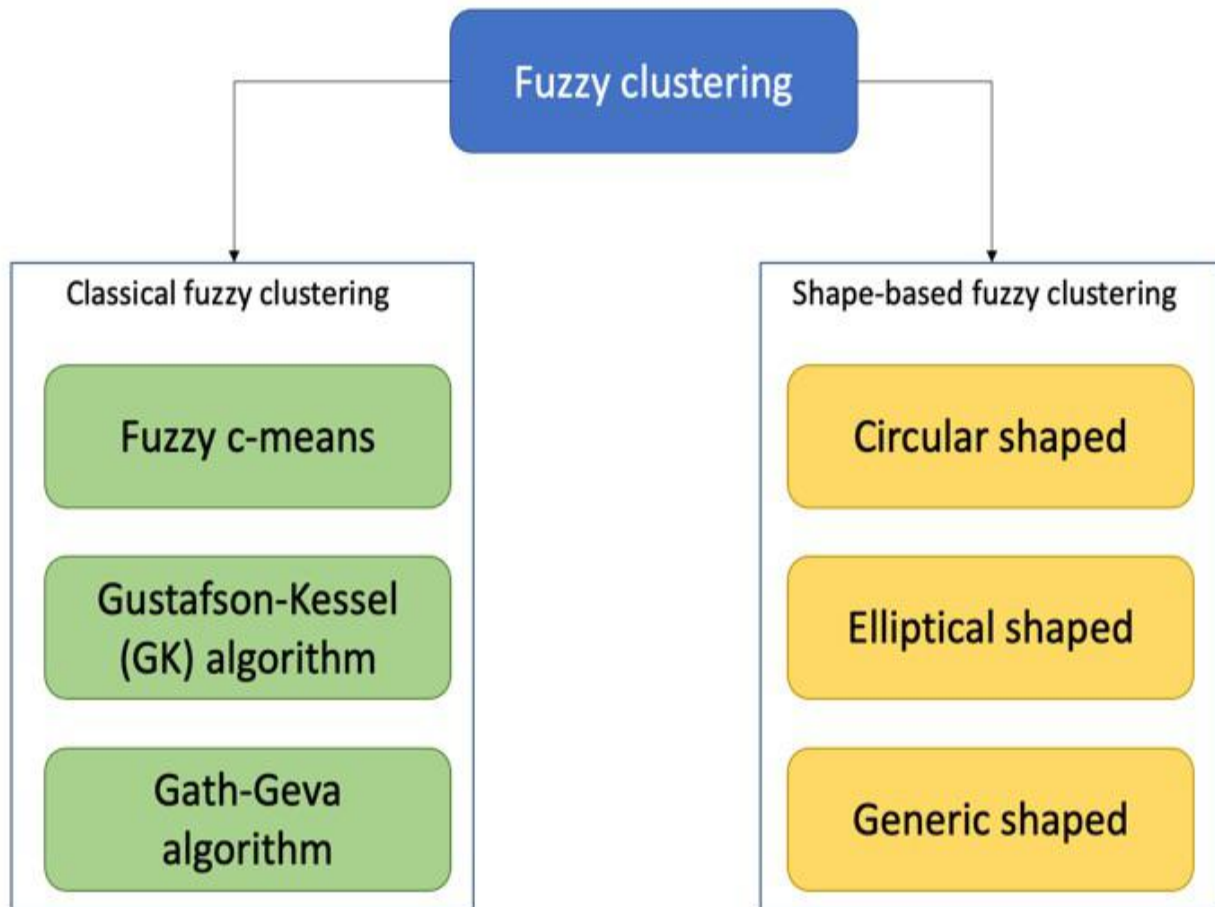
We will cover the Fuzzy c-means algorithm in detail here. Rest of the algorithms we will cover in brief.

1. *Gaustafson-Kessel algorithm* or sometimes called as GK algorithm works by associating an item with a cluster and a matrix. GL results in elliptical clusters and in order to modify as per varied structures in the datasets GK uses the covariance matrix. It allows the algorithm to capture the elliptical properties of the cluster. GK can result in narrower clusters and wherever the number of items is higher, those areas can be thinner.
2. Gath-Geva algorithm is not based on an objective function. The clusters can result in any shape since it is a fuzzification of statistical estimators.
3. The shape based clustering algorithms are self-explanatory as per their names. A circular fuzzy clustering algorithm will result in circular shaped clusters and so on.
4. Fuzzy c-means algorithm or FCM algorithm is the most popular fuzzy

clustering algorithm. It was initially developed in 1973 by J.C. Dunn and then it has been improved multiple times. It is quite similar to k-means clustering. There is a concept of membership which we will cover now.

Refer to Figure 5-14. In the first figure, we have some items or data points. These data points can be a part of a clustering dataset like customer transactions etc. In the second figure, we create a cluster for these data points. While this cluster is created, membership grades are allocated to each of the data points. These membership grades suggest the degree or the level to which a data point belong to a cluster. We will shortly examine the mathematical function to calculate these values.

We should not be confused between the degree and the probabilities. If we sum these degrees, we may not get 1 as these values are normalized between 0 and 1 for all the items.

In the third figure, we can observe and compare that the point 1, is closer to the cluster center and hence belong to the cluster to a higher degree than point 2 which is closer to the boundary or the edge of the cluster.

Figure 5-14 (i) We have some data points here which can be clustered (ii) The data points can be grouped into two clusters. For the first cluster, the cluster centroid is represented using a + sign. (iii) We can observe here that point 1 is much closer to the cluster center as compared to point 2. So, we can conclude that point 1 belongs to this cluster to a higher degree than cluster 2.
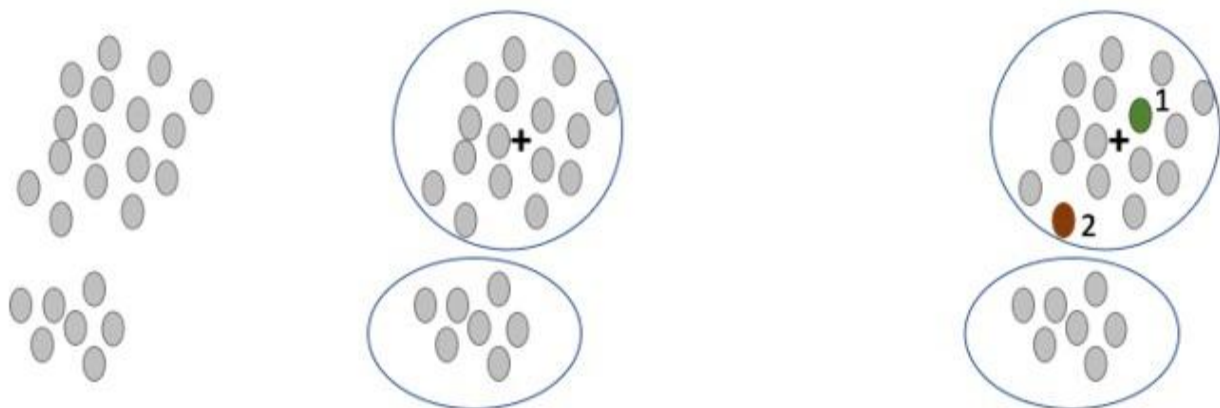
We will now venture into the technical details of the algorithm. It might be a little mathematically heavy though and hence this section can be treated as

optional.

Consider we have a set of n items

$$X = \{x_1, x_2, x_3, x_4, x_5 \ldots x_n\}$$

We apply FCM algorithm on these items. These n items are clustered into c fuzzy clusters based on some criteria. Let's say that we will get from the algorithm, a list of c cluster centers as $C = \{c_1, c_2, c_3, c_4, c_5 \ldots c_c\}$

The algorithm also returns a partition matrix which can be defined as below.

$$W = w_{i,j} \in [0,1], \ i = 1, \ldots, n, \ j = 1, \ldots, c$$

Here, each of the element $w_{i,j}$ is the degree to which each of the element in X belong to cluster c_j. This is the purpose of partition matrix.

Mathematically, we can get $w_{i,j}$ as shown in Equation 5-1. The proof of the equation is beyond the scope of the book.

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}.$$

(Equation 5-1)

The algorithm generates centroids for the clusters too. The centroid of a cluster is the mean of all the points in that cluster and the mean is weighted by their respective degrees of belonging to that cluster. wherein If we represent it mathematically, we can write it like in Equation 5-2.

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m},$$

In the Equation 5-1 and 5-2 we have a very important term "m". m is the hyperparameter which is used to control the fuzziness of the clusters. The values for m ≥ 1 and can be kept as 2 generally.

Higher the value of m, we will receive fuzzier clusters.

We will now examine step-by-step process in FCM algorithm:

1. First, we start as we start in k-means clustering. We choose the number of clusters we wish to have in the output.
2. Then the coefficients are allocated randomly to each of the data points.
3. Now we wish to iterate till the algorithm has converged. Recall how the k-means algorithm converges, wherein we initiate the process by randomly allocating the number of clusters. And then iteratively we calculate the centroid for each of the clusters. This is how kmeans converges. For FCM, we will utilizing the similar process albeit with slight differences. We have added a membership value $w_{i,j}$ and m.
4. For FCM, for the algorithm to converge we calculate the centroid for each of the cluster as per Equation 5-2.

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m},$$

5. For each of the data points, we also calculate its respective coefficient for being in that particular cluster. We will use Equation 5-1.
6. Now we have to iterate until FCM algorithm has converged. The cost function which we wish to minimize is given by the ()

$$\text{Objective function} = \operatorname*{arg\,min}_{C} \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^{m} \|\mathbf{x}_i - \mathbf{c}_j\|^2,$$

<div align="right">*(Equation 5-3)*</div>

Once this function has been minimized, we can conclude that that the FCM algorithm has converged. Or in other words, we can stop the process as the algorithm has finished processing.

It will be a good stage now to compare with kmeans algorithm. In kmeans, we have a strict objective function which will allow only one cluster membership while for FCM clustering, we can get different clustering membership based on the probability scores.

FCM is a very useful for the business cases where the boundary between clusters is not clear and stringent. Consider in the field of bio-informatics wherein a gene can belong to more than one cluster. Or if we have overlapping datasets like in fields of the marketing analytics, image segmentation etc. FCM can give comparatively more robust results than kmeans.

We will now proceed to Python implementation of FCM clustering in the next section.

🎯 POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

1. Fuzzy clustering allows us to create overlapping clusters. True or False.

2. A data point can belong to one and only one cluster. True or False.

3. If the value of "m" is lower, we get more clear clusters. True or False.

5.3.4 Python implementation of FCM

We have covered the process of FCM in the last section. We will now work on the Python implementation of FCM in this section.

Step 1: Import the necessary libraries.

```
import skfuzzy as fuzz
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Step 2: We will now declare a color palette, which will be used later for color coding the clusters.

```
color_pallete = ['r','m','y','c', 'brown', 'orange','m','k', 'gra
```

Step 3: We will define the cluster centers.

```
cluster_centers = [[1, 1],
          [2, 4],
          [5, 8]]
```

Step 4:

```
sigmas = [[0.5, 0.6],
          [0.4, 0.5],
          [0.1, 0.6]]
```

Step 5:

```
np.random.seed(5)
```

```
xpts = np.zeros(1)
ypts = np.zeros(1)
labels = np.zeros(1)
for i, ((xmu, ymu), (xsigma, ysigma)) in enumerate(zip(cluster_ce
    xpts = np.hstack((xpts, np.random.standard_normal(500) * xsig
    ypts = np.hstack((ypts, np.random.standard_normal(500) * ysig
    labels = np.hstack((labels, np.ones(500) * i))
```

Step 6:

```
fig0, ax0 = plt.subplots()
for label in range(5):
    ax0.plot(xpts[labels == label], ypts[labels == label], '.')
ax0.set_title('Data set having 500 points.')
plt.show()
```


Data set having 500 points.
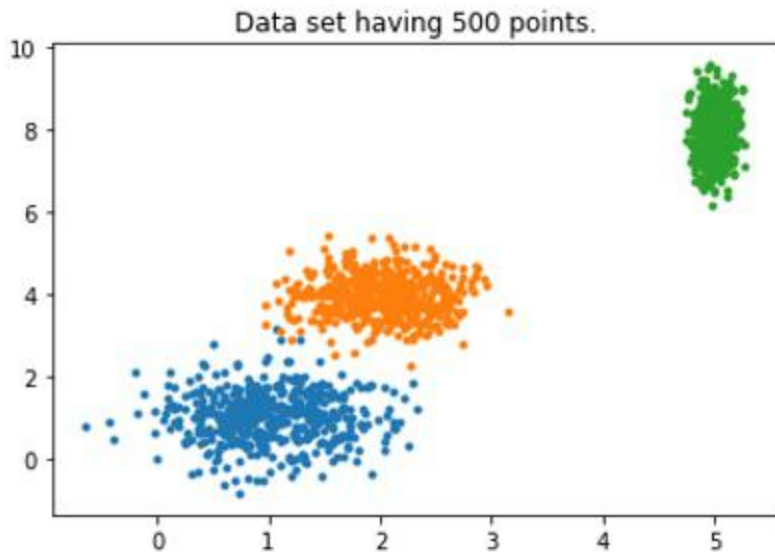
Step 7:

```
fig1, axes1 = plt.subplots(3, 3, figsize=(10, 10))
alldata = np.vstack((xpts, ypts))
fpcs = []

for ncenters, ax in enumerate(axes1.reshape(-1), 2):
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
        alldata, ncenters, 2, error=0.005, maxiter=1000, init=Non

    # Store fpc values for later
    fpcs.append(fpc)

    # Plot assigned clusters, for each data point in training set
    cluster_membership = np.argmax(u, axis=0)
    for j in range(ncenters):
        ax.plot(xpts[cluster_membership == j],
                ypts[cluster_membership == j], '.', color=colors[

    # Mark the center of each fuzzy cluster
    for pt in cntr:
        ax.plot(pt[0], pt[1], 'rs')

    ax.set_title('cluster_centers = {0}; FPC = {1:.2f}'.format(nc
```

```
    ax.axis('off')

fig1.tight_layout()
```


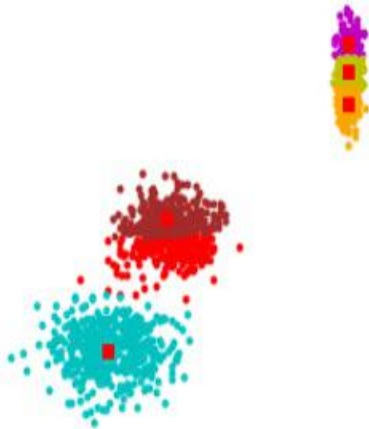cluster_centers = 2; FPC = 0.90   cluster_centers = 3; FPC = 0.91   cluster_centers = 4; FPC = 0.80

cluster_centers = 5; FPC = 0.74   cluster_centers = 6; FPC = 0.73   cluster_centers = 7; FPC = 0.65
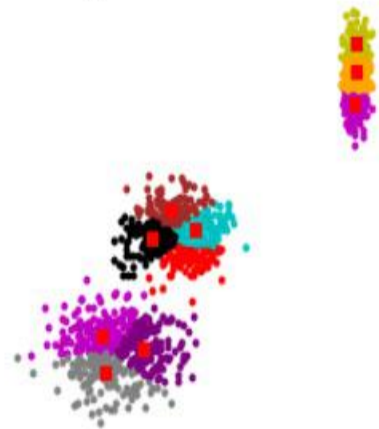
cluster_centers = 8; FPC = 0.63   cluster_centers = 9; FPC = 0.60   cluster_centers = 10; FPC = 0.59

With this we conclude Fuzzy Clustering and we can move to Gaussian Mixture model in the next section.

5.4 Gaussian Mixture Model

We would continue our discussion of soft clustering from the last section. Recall we introduced Gaussian Mixture Model there. Now we will elaborate on it. We will study the concept and have a Python implementation for it.

First let refresh our understanding of the *Gaussian distribution* or sometimes called as *normal distribution*. You might have heard bell-curve, it means the same thing.

In the Figure 5-15, observe that the distribution where the μ (mean) is 0 and σ^2 (standard deviation) is 1. It is a perfect normal distribution curve. Compare the distribution in different curves here.

(Image source – Wikipedia)

Figure 5-15 A Gaussian distribution is one of the most famous distributions. Observe how the values of mean and standard deviation are changed and their impact on the corresponding curve.
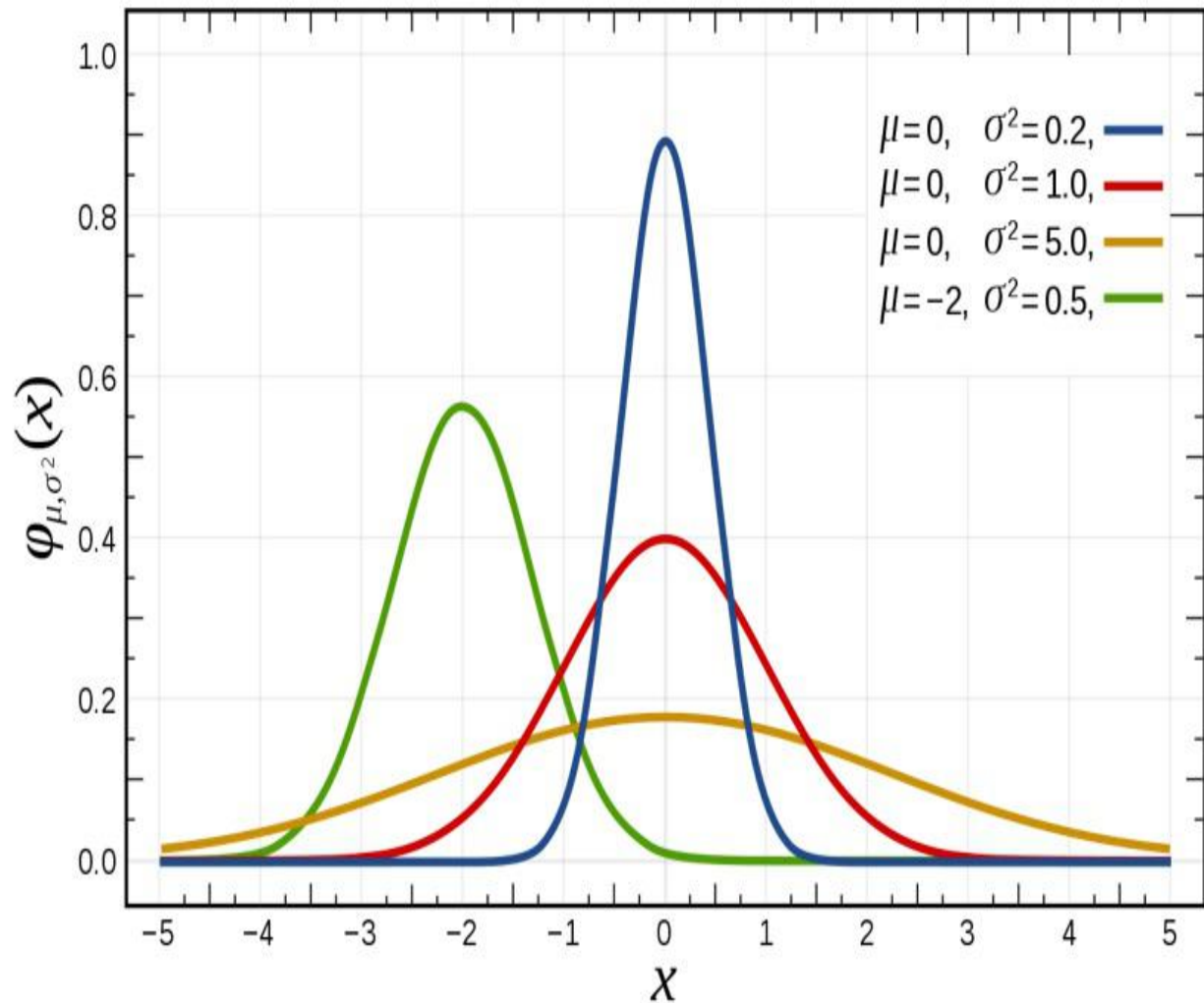
The mathematical expression for Gaussian distribution is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$f(x)$ = probability density function

σ = standard deviation

μ = mean

(Equation 5-4)

The equation above is also called the **Probability Density Function (pdf)**. In

the Figure 5-15, observe that the distribution where the μ is 0 and σ^2 is 1. It is a perfect normal distribution curve. Compare the distribution in different curves in the Figure 5-15 where by changing the values of mean and standard distribution, we are getting different graphs.

You might be wondering why we are using Gaussian distribution here. There is a very famous statistical theorem call as the *Central Limit Theorem*. We are explaining the theorem briefly here. As per the theorem, the more and more data we collect, the distribution tends to become more and more Gaussian. This normal distribution can be observed across all walks of like, be it chemistry, physics, mathematics, biology or any other branch. That is the beauty of Gaussian distribution.

The plot shown in Figure 5-15 is one-dimensional. We can have multi-dimensional Gaussian distribution too. In case of a multi-dimensional Gaussian distribution, we will get a 3-D figure shown in Figure 5-16. Our input was a scalar in one-dimensional. Now instead of scalar, out input is now a vector, mean is also a vector and represents the center of the data. And hence mean has the same dimensionality as the input data. The variance is now covariance matrix ∑. This matrix not only tells us the variance in the inputs, it also comments on the relationship between different variables. In other words, if the value of x is changed, how the values of y get impacted. Have a look at Figure 5-16 below. We can understand the relationship between x and y variable here.

(Image source – Wikipedia)

Figure 5-16 3-D representation of a Gaussian Distribution is shown here.
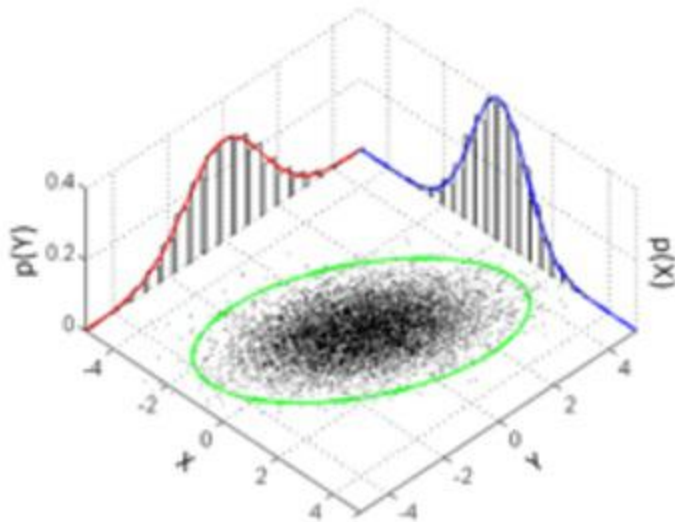
Covariance plays a significant role here. K-means does not consider the covariance of a dataset, which is used in GMM model.

Let's examine the process of GMM clustering. Imagine we have a dataset with *n* items. When we use GMM clustering, we do not find the clusters using centroid method, instead we fit a set of *k* gaussian distributions to the data set at hand. In other words, we have *k* clusters. We have to determine the parameters for each of these Gaussian distributions which are mean, variance and weight of a cluster. Once the parameters for each of the distribution are determined, then we can find the respective probability for each of the n items to belong to k clusters.

Mathematically, we can calculate the probability as shown in Equation 5-5. The equation is used to for us to know that a particular point x is a linear combination of k Gaussians. The term Φ_j is used to represent the strength of the Gaussian and it can be seen in the second equation that the sum of such strength is equal to 1.

$$p(x) = \sum_{j=1}^{k} \phi_j \mathcal{N}(x; \mu_j, \Sigma_j)$$

$$\sum_{j=1}^{k} \phi_j = 1$$

For spectral clustering, we have to identify the values of Φ, ∑ and μ. As you would imagine, getting the values of these parameters can be a tricky business. It is indeed a slightly complex called Expectation-Maximization technique or EM technique, which we will cover now. This section is quite heavy on mathematical concepts and is optional.

5.4.1 Expectation-Maximization (EM) technique

EM is a statistical and mathematical solution to determine the correct parameters for the model. There are quite a few techniques which are popular, perhaps maximum likelihood estimation is the most famous of them all. But at the same time, there could be a few challenges with maximum likelihood too. The data set might have missing values or in other words the dataset is incomplete. Or it is a possibility that a point in the dataset is generated by two different Gaussian distributions. Hence, it will be very difficult to determine that which distribution generated that data point. Here, EM can be helpful.

k-means uses only mean while GMM utilizes both mean and variance of the data.

The processes which are used to generate a data point are called *latent variables*. Since we do not know the exact values of these latent variables, EM firsts estimates the optimum values of these latent variables using the current data. Once this is done, then the model parameters are estimated. Using these model parameters, the latent variables are again determined. And using these new latent variables, new model parameters are derived. And the process continues till a good enough set of latent values and model

parameters are achieved which fit the data well. Let's study in more detail now.

We will take the same example we had in the last section.

Imagine we have a dataset with *n* items. When we use GMM clustering, we do not find the clusters using centroid method, instead we fit a set of *k* gaussian distributions to the data set at hand. In other words, we have *k* clusters. We have to determine the parameters for each of these Gaussian distributions which are mean, variance and weight of a cluster. Let's say that mean is μ_1, μ_2, μ_3, μ_4…. μ_k and covariance is Σ_1, Σ_2, Σ_3, Σ_4…. Σ_k. We can also have one more parameter to represent the density or strength of the distribution and it can be represented by the symbol Φ.

Now we will start with the Expectation or the E step. In this step, each data point is assigned to a cluster probabilistically. So, for each point we calculate its probability to belong to a cluster, if this value is high the point is in the correct cluster else the point is in the wrong cluster. In other words, we are calculating the probability that each data point is generated by each of the k Gaussians.

Since we are calculating probabilities, these are called soft assignments.

The probability is calculated using the formula in Equation 5-6. If we look closely, the numerator is the probability and then we are normalizing by the denominator. The numerator is the same we have seen in Equation 5-5.

$$W_j^{(i)} = \frac{\phi_j \mathcal{N}(x^{(i)}; \mu_j, \Sigma_j)}{\sum_{q=1}^{k} \phi_q \mathcal{N}(x^{(i)}; \mu_q, \Sigma_q)}$$

(Equation 5-6)

In the Expectation step above, for a data point $x_{i,j}$, where i is the row and j is the column, we are getting a matrix where rows are represented by the data points and columns are their respective Gaussian values.

Now the expectation step is finished, we will perform the maximization or the M step. In this step, we will update the values of μ, Σ and Φ using the formula below in Equation 5-7. Recall in k-means clustering, we simply take the mean of the data points and move ahead. We do something similar here albeit use the probability or the expectation we calculated in the last step.

The three values can be calculated using the Equation below. Equation 5-7 is the calculation of the covariances Σ_j, wherein we calculate the covariances of all the points, which is then weighted by the probability of that point being generated by Gaussian j. The mathematical proofs are beyond the scope of this book.

$$\Sigma_j = \frac{\sum_{i=1}^{N} W_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{N} W_j^{(i)}}$$

(Equation 5-7)

The mean μ_j, is determined by Equation 5-8. Here, we determine the mean for all the points, weighted by the probability of that point being generated by Gaussian j.

$$\mu_j = \frac{\sum_{i=1}^{N} W_j^{(i)} x^{(i)}}{\sum_{i=1}^{N} W_j^{(i)}}$$

(Equation 5-8)

Similarly, the density or the strength is calculated by Equation 5-9, where we add all the probabilities for each point to be generated by Gaussian j and then divide by the total number of points N.

$$\phi_j = \frac{1}{N} \sum_{i=1}^{N} W_j^{(i)}$$

Based on these values, new values for $\sum$, μ and Φ are derived, and the process continues till the model converges. We stop when we are able to maximize the log-likelihood function.

It is a complex mathematical process. We have covered it to give you the in-depth understanding of what happens in the background of the statistical algorithm. The Python implementation is much more straight forward than the mathematical concept which we will cover now.

◎ POP QUIZ – answer these question to check your understanding..
Answers at the end of the book

1. Gaussian distribution has mean equal to 1 and standard deviation equal to 0. True or False.

2. GMM models does not consider the covariance of the data. True or False.

5.4.2 Python implementation of GMM

We will first import the data and then we will compare the results using kmeans and GMM.

Step 1: We will import all the libraries and import the dataset too.

```
import pandas as pd
data = pd.read_csv('vehicle.csv')
import matplotlib.pyplot as plt
```

Step 2: We will now drop any NA from the dataset.

```
data = data.dropna()
```

Step 3: We will now fit a `kmeans` algorithm. We are keeping the number of clusters as 5. Please note that we are not saying that they are idle number of clusters. These number of clusters are only for illustrative purpose. We are declaring a variable kmeans and then using five clusters and then the dataset is fit next.

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5)
kmeans.fit(data)
```

Step 4: We will now plot the clusters. First, a prediction is made on the dataset and then the values are added to the data frame as a new column. The data is then plotted with different colors representing different clusters.

The output is shown in the plot below.

```
pred = kmeans.predict(data)
frame = pd.DataFrame(data)
frame['cluster'] = pred

color=['red','blue','orange', 'brown', 'green']
for k in range(0,5):
    data = frame[frame["cluster"]==k]
    plt.scatter(data["compactness"],data["circularity"],c=color[k
plt.show()
```
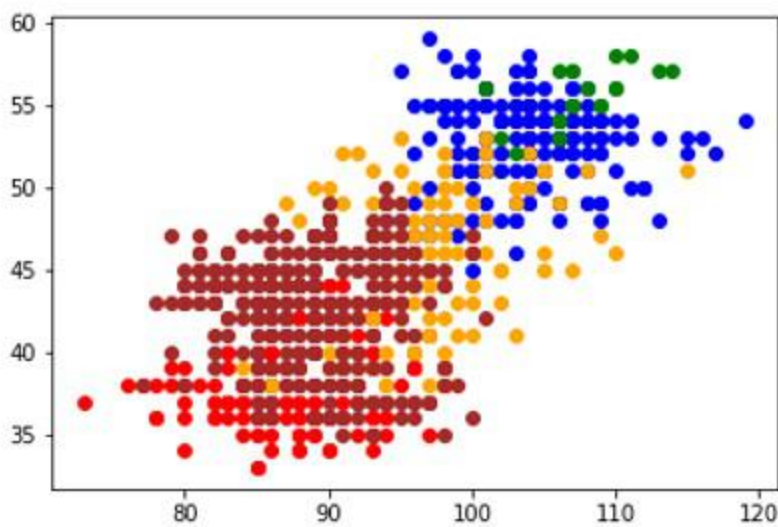
Step 4: We will now fit a GMM model. Note that the code is the same as the kmeans algorithm only the algorithm's name has changed from kmeans to GaussianMixture.

```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=5)
gmm.fit(data)

#predictions from gmm
labels = gmm.predict(data)
frame = pd.DataFrame(data)
frame['cluster'] = labels
```

Step 5: We will now plot the results. The output is shown below.

```
color=['red','blue','orange', 'brown', 'green']
for k in range(0,5):
    data = frame[frame["cluster"]==k]
    plt.scatter(data["compactness"],data["circularity"],c=color[k
plt.show()
```
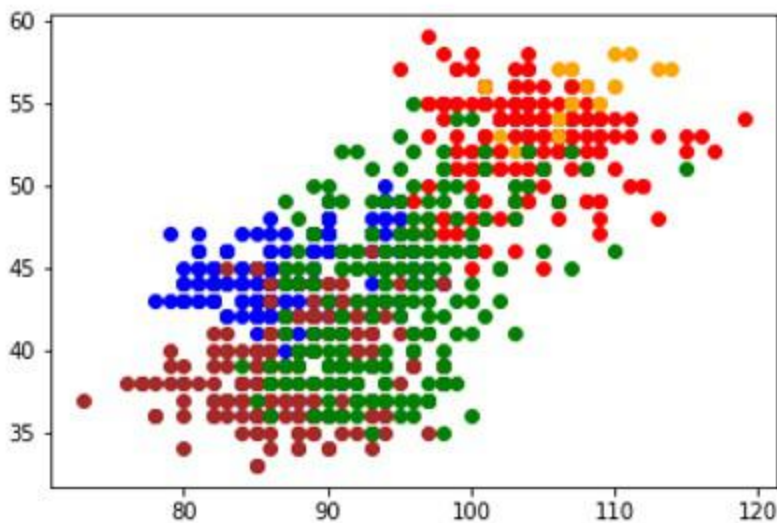

Step 6: You are advised to run the code with different values of clusters to observe the difference. In the plots below, the left one is kmeans with two clusters while the right is GMM with two clusters.
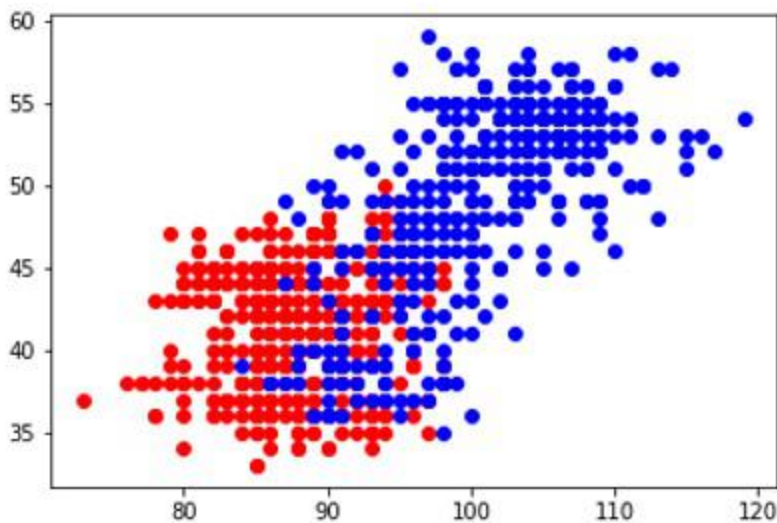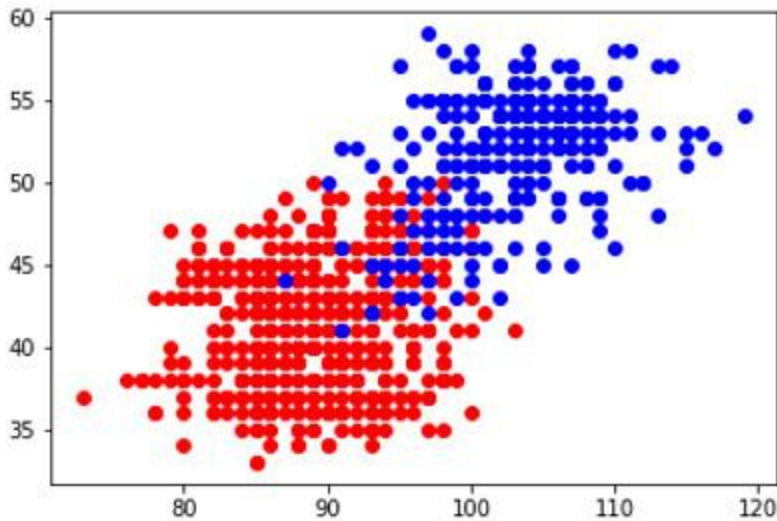
Gaussian distribution is one of the most widely used data distribution used. If we compare kmeans and GMM model, we would understand that kmeans does not consider the normal distribution of the data. The relationship of various data points is also not considered in kmeans.

Kmeans is a distance-based algorithm, GMM is a distribution based algorithm.

In short, it is advantageous to use GMM models for creating the clusters particularly when we have overlapping datasets. It is a useful technique for financial and price modelling, NLP based solutions etc.

With this, we have covered all the algorithms in the chapter. We can now move to the summary.

5.5 Summary

In this chapter, we have explored three complex clustering algorithms. You might have felt the mathematical concepts a bit heavy. They are indeed heavy but give a deeper understanding of the process. It is not necessary that these algorithms are the best ones for each and every problem. Ideally, in the real-world business problem we should first start with classical clustering algorithms – kmeans, hierarchical and DBSCAN. If we do not get acceptable results then, we can try the complex algorithms.

Many times, a data science problem is equated to the choice of algorithm, which it is not. The algorithm is certainly an important ingredient of the entire solution, but it is not the only one. In the real-world datasets, there are a lot of variables and the amount of data is also quite big. The data has a lot of noise. We have to account for all these factors when we shortlist an algorithm. Algorithm maintenance and refresh is also one of the major questions we have in mind. All these finer points are covered in much detail in the last chapter of the book.

We will cover complex dimensionality reduction techniques in the next chapter. You can move to questions now.

Practical next steps and suggested readings

1. In chapter 2 we have done clustering using various techniques. Use the datasets from there and perform Spectral clustering, GMM and FCM clustering to compare the results.
2. There are datasets provided at the end of chapter 2 which can be used for clustering.
3. Get the credit card dataset for clustering from this Kaggle link (https://www.kaggle.com/vipulgandhi/spectral-clustering-detailed-explanation) and from the famous IRIS dataset which we have used earlier too.
4. There is a great book Computational Network Science by Henry

Hexmoor to study the mathematical concepts.

5. Get Spectral clustering papers from the links below and study them:
6. On spectral clustering: analysis and an algorithm
https://proceedings.neurips.cc/paper/2001/file/801272ee79cfde7fa596057
Paper.pdf
7. Spectral clustering with eigenvalue selection
http://www.eecs.qmul.ac.uk/~sgg/papers/XiangGong-PR08.pdf
8. The mathematics behind spectral clustering and the equivalence to PCA
https://arxiv.org/pdf/2103.00733v1.pdf
9. Get GMM papers from the link below and explore them:
10. A particular Gaussian Mixture Model for clustering
https://citeseerx.ist.psu.edu/viewdoc/download?
doi=10.1.1.79.7057&rep=rep1&type=pdf
11. Application of Compound Gaussian Mixture Model in the data stream
https://ieeexplore.ieee.org/document/5620507
12. Get FCM papers from the link below and study them:
13. FCM: the fuzzy c-means clustering algorithm
https://www.sciencedirect.com/science/article/pii/0098300484900207
14. A survey on Fuzzy c-means clustering techniques
https://www.ijedr.org/papers/IJEDR1704186.pdf
15. Implementation of Fuzzy C-Means and Possibilistic C-Means Clustering
Algorithms, Cluster Tendency Analysis and Cluster Validation
https://arxiv.org/pdf/1809.08417.pdf

6 Dimensionality reduction (advanced)

"Life is really simple, but we insist on making it complicated - Confucius"

Simplicity is a virtue. Both in life and in data science. We have discussed a lot of algorithms so far – a few of them are simple enough and some of them are a bit complicated. In Part one of the book, we studied simpler clustering algorithms and in the last chapter, we examined advanced clustering algorithms. Similarly, we studied a few dimensionality algorithms like PCA in chapter 3. Continuing on the same note, we will study two advanced dimensionality reduction techniques in this chapter.

The advanced topics we are covering this part and the next part of the book are meant to prepare you for complex problems. Whilst you can apply these advanced solutions, it is always advisable to start with the classical solution like PCA for dimensionality reduction. And if the solution achieved it not at par, then you can try the advanced solutions.

Dimensionality reduction is one of the most sought-after solution particularly when we have a large number of variables. Recall "Curse of Dimensionality" we discussed in chapter 3. You are advised to refresh chapter 3 before moving forward. We will cover t-distributed Stochastic Neighbour Embedding (t-SNE) and Multidimensional Scaling (MDS) in this chapter. This chapter will have some mathematical concepts which create the foundation of the advanced techniques we are going to discuss. As always, we will have the concept discussion followed by Python implementation. We will have a short case study at the end of the chapter. And, in this chapter we are developing a solution using images dataset too!

There can be a dilemma in your mind. What is the level of mathematics required and is an in-depth statistical knowledge is a pre-requisite? The answer is both Yes and No. Whilst, having a mathematical understanding will allow you to understand the algorithms and appreciate the process in greater

depth; at the same time for real-world business implementation sometimes one might want to skip the mathematics and directly move to the implementation in Python. We would suggest to have at least more than basic understanding of the mathematics to fully grasp the concept. In this book, we are providing that level of mathematical support without going in too much depth – an optimal mix of practical world and mathematical concepts.

In this sixth chapter of the book, we are going to cover the following topics:

1. t-distributed Stochastic Neighbour Embedding (t-SNE)
2. Multidimensional Scaling (MDS)
3. Python implementations of the algorithms
4. Case study

Welcome to the sixth chapter and all the very best!

6.1 Technical toolkit

We will continue to use the same version of Python and Jupyter notebook as we have used so far. The codes and datasets used in this chapter have been checked-in at https://github.com/vverdhan/UnsupervisedLearningWithPython/tree/main/Cha location.

You would need to install Keras as an additional Python libraries in this chapter . Along with this we will need the regular libraries – numpy, pandas, matplotlib, seaborn, sklearn. Using libraries, we can implement the algorithms very quickly. Otherwise, coding these algorithms is quite a time-consuming and painstaking task.

Let's get started with Chapter 6 of the book!

6.2 Multidimensional Scaling (MDS)

I love to travel. Unfortunately, due to the COVID pandemic, the travelling has taken a hit. As you know, maps prove to be quite handy while travelling. Now, imagine you are given a task. You receive distances between some

cities around the world. For example, between London and New York, London and Paris, Paris and New Delhi and so forth. And then we ask you to recreate the map from which these distances have been derived. If we have to recreate that two-dimensional map, that will be through trial and error, we will make some assumptions and move ahead with the process. It will surely be a tiring exercise prone to error and quite time consuming indeed. MDS can do this task easily for us.

While thinking of the above example, ignore the fact that earth is not flat. And assume that the distance measurement metric is constant. For example, there is no confusion in miles or kilometres.

As an illustration, consider the Figure 6-1.

Figure 6-1 Illustration of distance between the cities and if they are represented on a map. The figure is only to help develop an understanding and does not represent the actual results.

Formally put, if we have x data points, multidimensional scaling (MDS) can help us in converting the information of the pairwise distance between these x points to a configuration of points in a Cartesian space. Or simply put, MDS transforms a large dimensional dataset into a lower dimensional one and in the process keeping the distance or the similarity between the points same.

To simplify, consider the image below. Here we have three points - A, B and C. We are representing these points in a 3D space. And then we are representing the three points in a 2D space and finally they are represented in a 1D space. The distance between the points is not up to scale in the diagrams

below in Figure 6-2. The example shown in Figure 6-2 represents the meaning of lowering the number of dimensions.

Figure 6-2 Representation of three points – first we are showing three points in a three-dimensional space. Then they are being represented in a 2D space and then finally in a single dimensional space.
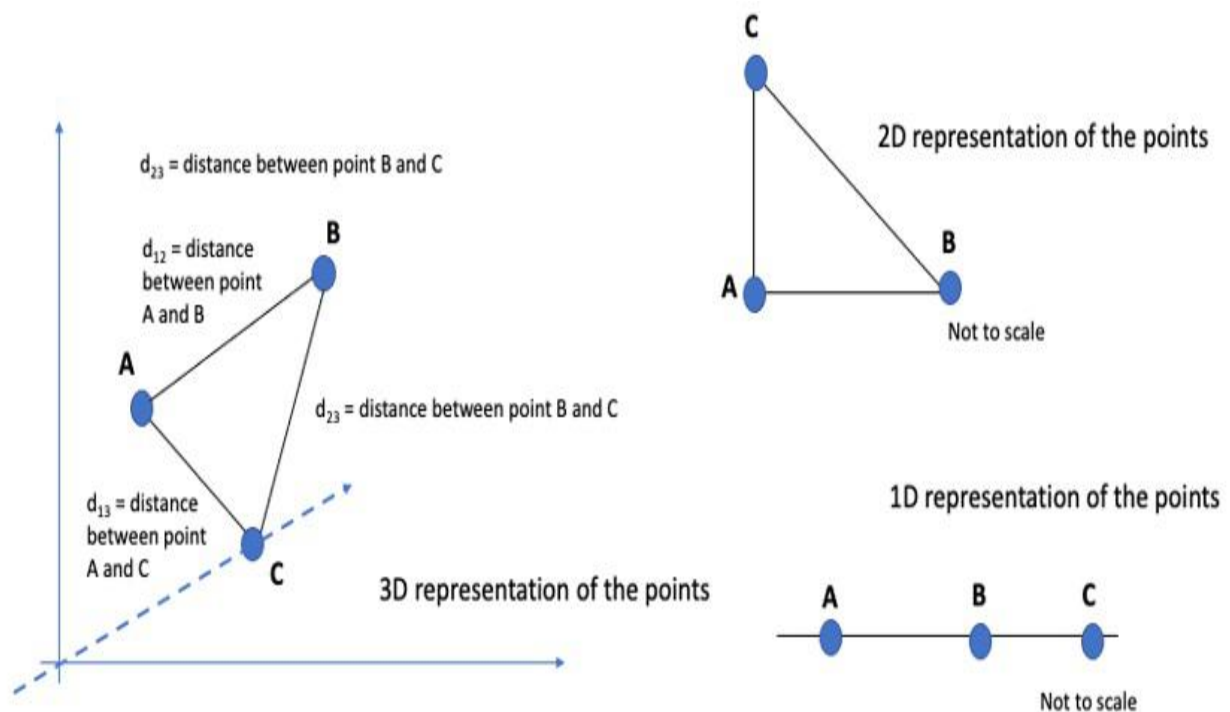
Hence, in MDS a multidimensional data is reduced to lower number of dimensions.

We can have three types of MDS algorithms

1. Classical MDS,
2. Metric multidimensional scaling and
3. Non-metric multidimensional scaling.

We will examine metric MDS process in detail in the book while we will cover the classical and non-metric briefly.

Imagine we have two points – i and j. Let us assume that the original distance between two points is d_{ij} and the corresponding distance in the lower

dimensional space is d_{ij}.

In classical MDS, the distances between the points are treated as Euclidean distances and the original and fitted distances are represented in the same metric. It means that if the original distances in higher dimensional space are calculated using Euclidean method, the fitted distances in lower dimensional are also calculated using Euclidean distance. We already know how to calculate Euclidean distances. For example, we have to find the distance between points i and j and let's say the distance is d_{ij}. The distance can be given by the Euclidean distance formula given by Equation 6-1.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

(Equation 6-1)

Recall in earlier chapters, we have discussed other distance functions like Manhattan, Hamming distance etc. You are advised to refresh them.

We will now come to non-metric MDS. We just now noted that Euclidean distance can be used to calculate the distance between two points. Sometimes it is not possible to take the actual values of the distances like when d_{ij} is the result of an experiment where subjective assessments were made. Or in other words, where a rank was allocated to the various data parameters. For example, if the distance between point 2 and 5 was at rank 4 in the original data, in such a scenario, it will not be wise enough to use absolute values of d_{ij} and hence relative values or *rank-values* have to be used. This is the process in non-metric MDS. For example, imagine we have four points – A, B, C and D. We wish to rank the respective distances between these four points. The respective combinations of points can be – A and B, A and C, A and D, B and C, B and D, and final C and D. Their distances can be ranked as shown in the Table 6-1.

Table 6-1 Representing the respective distance between four points and the ranks of the distances

Pair of the points	Distance	Ranks of the respective distances

A and B	100	3
A and C	105	4
A and D	95	2
B and C	205	6
B and D	150	5
C and D	55	1

So, in non-metric MDS method, instead of using the actual distances we use the respective ranks of the distance. We will now move to metric MDS method.

We know that in classical MDS, the original and fitted distances are represented in the same metric. In *metric MDS*, it is assumed that the values of d_{ij} can be transformed into Euclidean distances by employing some parametric transformation on the datasets. In some articles, you might find classical and metric MDS to be used interchangeably.

In MDS, as a first step, the respective distances between the points are calculated. Once the respective distances have been calculated, then MDS will try to represent the higher dimensional data point into lower dimensional space. To perform this, an optimization process has to be carried so that the most optimum number of resultant dimensions can be chosen. And hence, a loss function or cost function has to be optimized.

If you do not know what is a cost function, go through this section below.

Cost function

We use algorithms to predict the values of a variable. For example, we might use some algorithm to predict the expected demand of a product next year. We would want the algorithm to predict as much accurate as possible. Cost functions are a simple method to check the performance of the algorithms.

Cost function is a simple technique to measure the effectiveness of our algorithms. It is the most common method used to gauge the performance of a predictive model. It compares the original values and the predicted values by the algorithm and calculates how wrong the model is in its prediction.

As you would imagine, in an ideal solution, we would want the predicted values to be the same as the actual values, which is very difficult to achieve. If the predicted values differ a lot from the actual values, the output of a cost function is higher. If the predicted values are closer to the actual values, then the value of a cost function is lower. A robust solution is one which has a lower value of the cost function. Hence, the objective to optimize any algorithm will be to minimize value of the cost function. Cost function is also referred as *loss function,* these two terms can be used interchangeably.

In metric MDS, we can also call the cost function as **Stress**. The formula for Stress is given by Equation 6-2 as given below:

$$\text{Stress}_D(x_1, x_2, \ldots, x_N) = \left(\sum_{i \neq j = 1, \ldots, N} \left(d_{ij} - \|x_i - x_j\| \right)^2 \right)^{1/2}$$

(Equation 6-2)

Let's understand the equation now:

1. Term Stress_D is the value MDS function has to minimize.
2. The data points with the new set of coordinates in a lower dimensional space are represented by x, x_2, x_3…. x_N.
3. The term $\|x_i - x_j\|$ is the distance between two points in their lower dimensional space.

4. The term d_{ij} is original distance between the two points in the original multi-dimensional space.

By looking at the equation, we can clearly understand that if the values of $\|x_i - x_j\|$ and d_{ij} are close to each other, the value of the resultant stress will be small.

Minimizing the value of stress is the objective of the loss function.

To optimize this loss function, multiple approaches can be used. One of the most famous method is using a gradient descent which was originally proposed by Kruskal and Wish in 1978. The gradient descent method is very simple to understand and can be explained using a simple analogy.

Imagine you are standing on top of a mountain and you want to get down. While doing so, you want to choose the fastest path because you want to get down as fast as possible (no, you cannot jump!). So, to take the first step, you will look around and whichever is the steepest path, you can take a step in that direction and you will reach a new point. And then again, you will take a step in the steepest direction. We are showing that process in Figure 6-3(i).

Figure 6-3 (i) The first figure is of a person standing on top of a mountain and trying to get down. The process of gradient descent follows this method (ii) The actual process of optimization of a cost function in gradient descent process. Note that at the point of convergence, the value of the cost function is minimum.
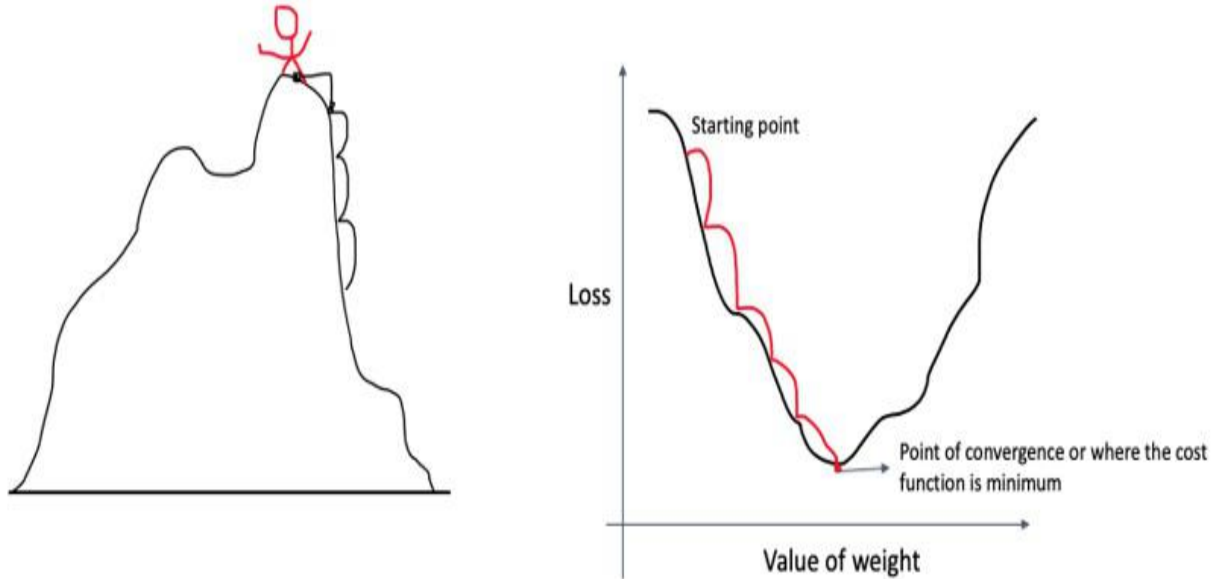
Now, if an algorithm has to achieve the similar feat, the process can be represented in Figure 6-3 (ii), wherein a loss function starts at a point and finally reaches the Point of convergence. At this point of convergence, the cost function is minimum.

MDS differs from the other dimensionality reduction techniques. As compared to techniques like PCA, MDS does not make any assumptions about the data set and hence can be used for a larger type of datasets. Moreover, MDS allows to use any distance measurement metric. Unlike PCA, MDS is not an eigenvalue-eigenvector technique. Recall in PCA, the first axis captures the maximum amount of variance, second axis has the next best variance and so on. In MDS, there is no such condition. The axes in MDS can be inverted or rotated as per the need. Next, in most of the other dimensional reduction methods used, the algorithms do calculate a lot of axes but they cannot be viewed. In MDS, smaller number of dimensions are explicitly chosen at the start. And hence, there is less ambiguity in the solution. Further, in other solutions generally there is only one unique solution whereas MDS tries to iteratively find the acceptable solution. It means in MDS there can be multiple solutions for the same dataset.

But at the same time, the computation time required for MDS is higher for bigger datasets. And there is a catch in the Gradient Descent method used for optimization. Refer to Figure 6-4. Let's refer to the mountain example we covered in the last section. Imagine that while you are coming down from the

top of the mountain. The starting point is A and the bottom of the mountain is at point C. While you are coming down, you reached point B. As you can see in Figure 6-4(i), there is a slight elevation around point B. At this point B, you might incorrectly conclude that you have reached the bottom of the mountain. In other words, you will think that you have finished your task. This is the precise problem of the local minima.

It is a possibility that instead of global minima, the loss function might be stuck in a local minima. The algorithm might think that it has reached the point of convergence, while the complete convergence might not have been achieved still and we are at local minima.

Figure 6-4 While the first figure is the point of convergence and represents the gradient descent method, note that in the second figure the global minima is somewhere else, while the algorithm can be stuck at a local minima. The algorithm might believe that it has optimized the cost function and reached the point of global minima whereas, it has only reached the local minima.
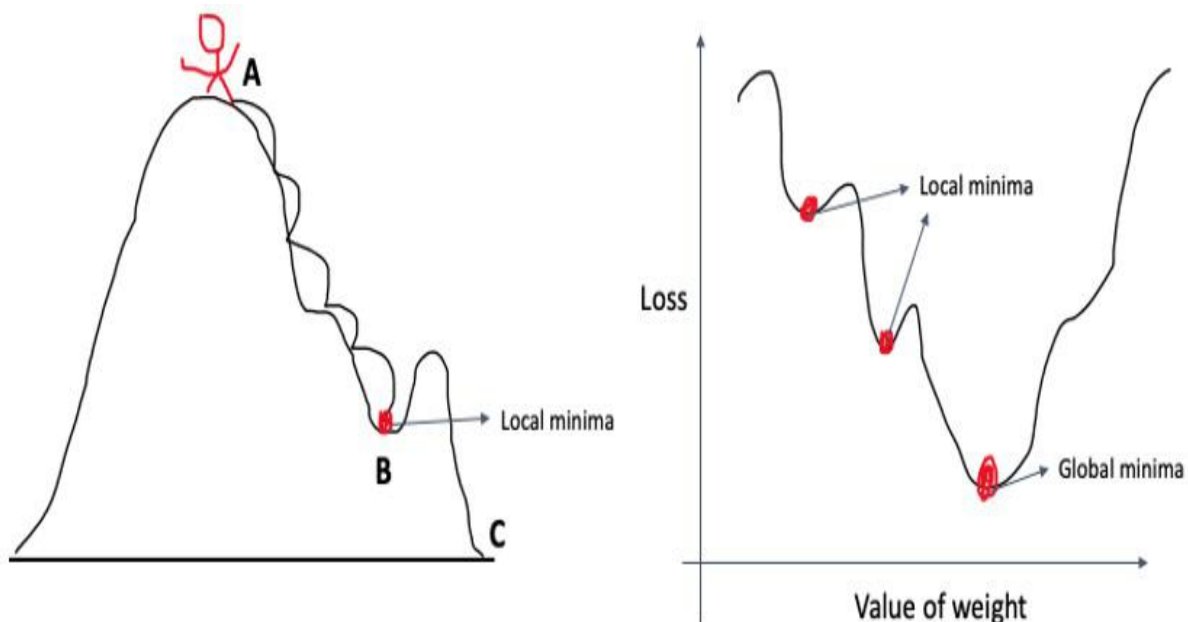
There is still a question to be answered about the efficacy of the MDS solution. How can we measure the effectiveness of the solution? In the original paper, Kruskal has recommended about the stress values to measure the goodness-of-fit of the solution which are shown in Table 6-1. The recommendations are mostly based on empirical experience of Kruskal. These stress values are based on Kruskal's experience.

Stress Values	Goodness-of-fit
0.200	Poor
0.100	Fair
0.050	Good
0.025	Excellent
0.000	Perfect

The next logical question is – how many final dimensions we should choose? Scree plot provides the answer as shown in Figure 6-5. Recall, in chapter 2 we used the similar elbow method to choose the optimal number of clusters in kmeans clustering. For MDS too, we can have the elbow method to determine the most optimal number of components to represent the data.

Figure 6-5 Scree plot to find the most optimal number of components. It is similar to the kmeans solution we have discussed in the earlier chapters; we have to look for the elbow in the plot.

Selecting k with the Elbow Method
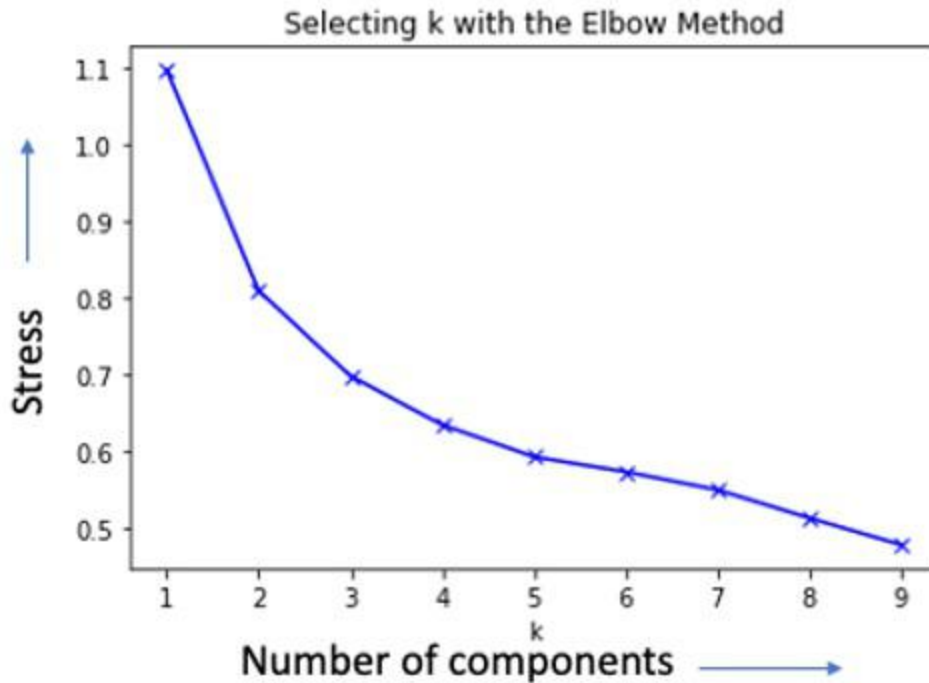
This concludes our discussion on MDS. We will now move to the Python implementation of the algorithm.

🎯 POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

1. What is the difference between metric and non-metric MDS algorithm?

2. Gradient descent is used to maximise the cost. True or False.

3. Explain gradient descent method using a simple example.

6.2.1   Python implementation of MDS

We will now have the Python implementation of MDS method. We will use the famous Iris dataset which we have used previously too. The implementation of the algorithm is quite simple, thanks to the libraries available in the scikit learn package.

The implementation is generally simple as the heavy lifting is done by the libraries.

Step 1: We will first load the libraries. The usual suspects are `sklearn`, `matplotlib`, `numpy` and we also load `MDS` from `sklearn`.

```
import numpy as np
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
from sklearn.manifold import MDS
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

Step 2: Load the data set now. Iris dataset is available in the `sklearn` library so we need not import excel or .csv file here.

```
raw_data = load_iris()
dataset = raw_data.data
```

Step 3: A requirement for MDS is that the dataset should be scaled before the actual visualization is done. We are using `MixMaxScalar()` function to achieve the same. MinMax scaling simply scales the data using the formula below:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
d_scaler = MinMaxScaler()
dataset_scaled = d_scaler.fit_transform(dataset)
```

As an output of this step, the data is scaled and ready for the next step of modelling.

Step 4: We now invoke the MDS method from sklearn library. The random_state value allows us to reproduce the results. We have decided the number of components as 3 for the example.

```
mds_output = MDS(3,random_state=5)
```

Step 5: We will now fit the scaled data created earlier using the MDS model.

```
data_3d = mds_output.fit_transform(dataset_scaled)
```

Step 6: We are now declaring the colors we wish to use for visualization. And next, the data points are visualized in a scatter plot.

```
mds_colors = ['purple','blue', 'yellow']
for i in np.unique(raw_data.target):
  d_subset = data_3d[raw_data.target == i]

  x = [row[0] for row in d_subset]
  y = [row[1] for row in d_subset]
  plt.scatter(x,y,c=mds_colors[i],label=raw_data.target_names[i])
plt.legend()
plt.show()
```

The output of the code above can be shown below in Figure 6-6:

Figure 6-6 Output for the IRIS data
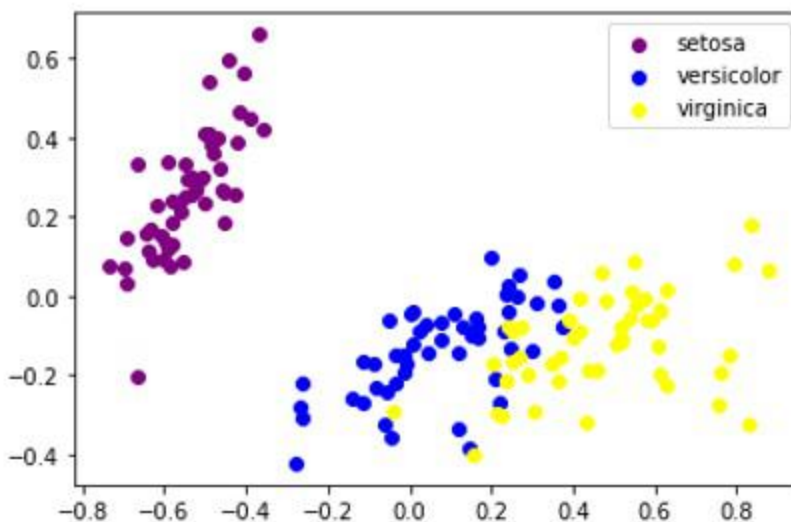
The above example of Python implementation is a visualization of the IRIS data. It is quite simple example but it does not involve *stress* and optimization for the number of components. *We* will now work on a curated dataset to implement MDS.

Distance	A	B	C	D	E
A	0	40	50	30	40
B	40	0	40	50	20
C	50	40	0	20	50
D	30	50	20	0	20
E	40	20	50	20	0

Let us assume we have five cities and the respective distance between them is given in Table 6-2.

Step 1: We have already imported the libraries in the last code.

```
import numpy as np
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
from sklearn.manifold import MDS
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

Step 2: Let's now create the dataset. We are creating the dataset here, but in real business scenarios it will be in the form of distances only.

```
data_dummy_cities = {'A':[0,40,50,30,40],
        'B':[40,0,40,50,20],
        'C':[50,40,0,20,50],
        'D':[30,50,20,0,20],
        'E':[40,20,50,20,0],
        }
cities_dataframe = pd.DataFrame(data_dummy_cities, index =['A','B
cities_dataframe
```

```
1  cities_dataframe
```

	A	B	C	D	E
A	0	40	50	30	40
B	40	0	40	50	20
C	50	40	0	20	50
D	30	50	20	0	20
E	40	20	50	20	0

Step 3: We will now use the MinMaxScalar() function to scale the dataset as we did in the last coding exercise.

```
scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(cities_dataframe)
```

Step 4: Now, let's work towards finding the most optimal number of components. We will iterate for different values of number of components. For each of the value of number of components, we will get the value of stress. And at a point, where a kink is observed, that is the most optimal number of components.

As a first step, we will declare an empty dataframe which can be used to store the values of number of components and corresponding stress values. Then, we are iterating from 1 to 10 in a for loop. And finally, for each of the values of components (1 to 10), we get the respective values of stress.

```
MDS_stress = []
for i in range(1, 10):
    mds = MDS(n_components=i)
    pts = mds.fit_transform(df_scaled)
    MDS_stress.append(mds.stress_)
```

Step 5: We have got the values of stress. We will now plot these values in a graph. The respective labels for each of the axes are also given. Look at the kink at values 2 and 3. This can be the values of optimal values of number of

components.

```
plt.plot(range(1, 10), MDS_stress)
plt.xticks(range(1, 5, 2))
plt.title('Plot of stress')
plt.xlabel('Number of components')
plt.ylabel('Stress values')
plt.show()
```

Figure 6-7 Scree plot to select the optimized number of components
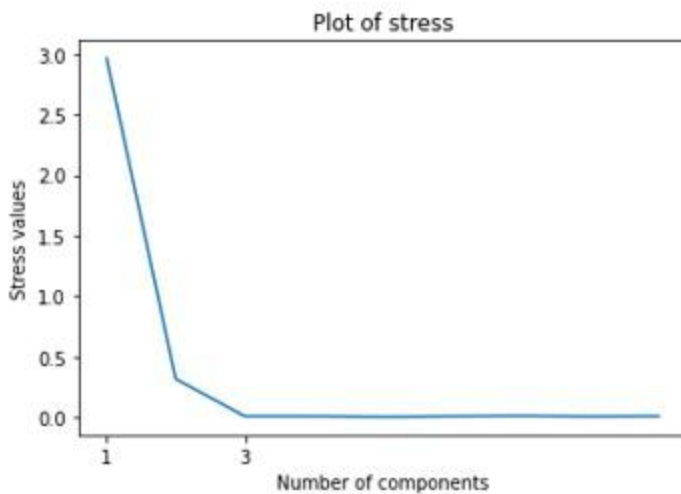
Step 6: We will now run the solution for number of components = 3. If we look at the values of stress, number of components = 3, it generates the minimum values of stress as 0.00665.

```
mds = MDS(n_components=3)
x = mds.fit_transform(df_scaled)
cities = ['A','B','C','D','E']

plt.figure(figsize=(5,5))
plt.scatter(x[:,0],x[:,1])
plt.title('MDS with Sklearn')
for label, x, y in zip(cities, x[:, 0], x[:, 1]):
    plt.annotate(
        label,
        xy = (x, y),
        xytext = (-10, 10),
        textcoords = 'offset points'
    )
plt.show()
```

```
print(mds.stress_)
```

Figure 6-8 Output for the MDS dataset, representation of the 5 cities in a plot
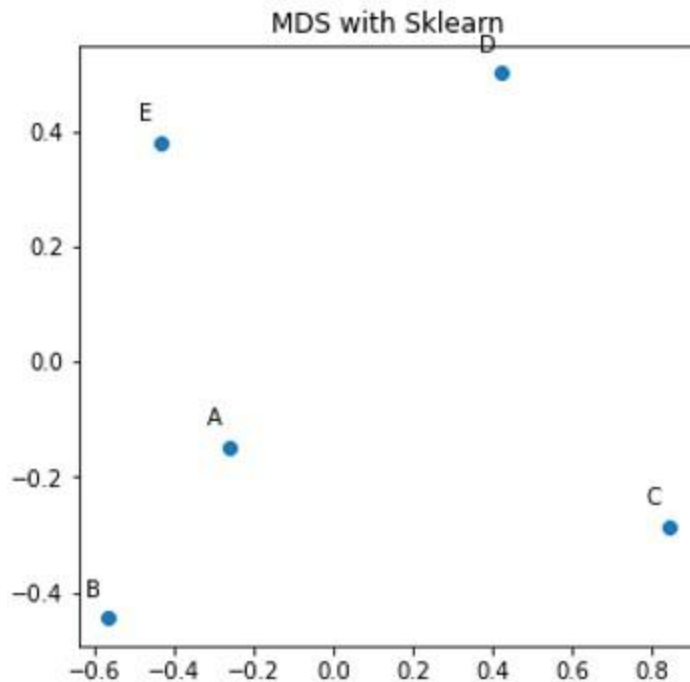
This concludes our section on MDS algorithm. We discussed the foundation and concepts, pros and cons, algorithm assessment and Python implementation of MDS. It is a great solution for visualization and dimensionality reductions. It is one of the non-linear dimensionality reductions methods.

We will now move to t-SNE, second dimensionality reduction methods in this chapter.

6.3 t-distributed stochastic neighbor embedding (t-SNE)

If a data set is really high dimensional, the analysis becomes cumbersome. The visualization is even more confusing. We have covered that in great detail in Curse of Dimensionality section in Chapter 2. You are advised to revisit the concept before proceeding.

One such really high-dimensional dataset can be image data. We find it difficult to comprehend such data which is really high-dimensional.

You would have used facial recognition software in your smartphones. For such solutions, facial images have to analyzed and machine learning models have to be trained. Look at the pictures below in Figure 6-9– we have a human face, a bike, a vacuum cleaner and screen capture of a game.

Figure 6-9 Images are quite complex to decipher by an algorithm. Images can be of any form and can be of a person, or an equipment or even any game screen.
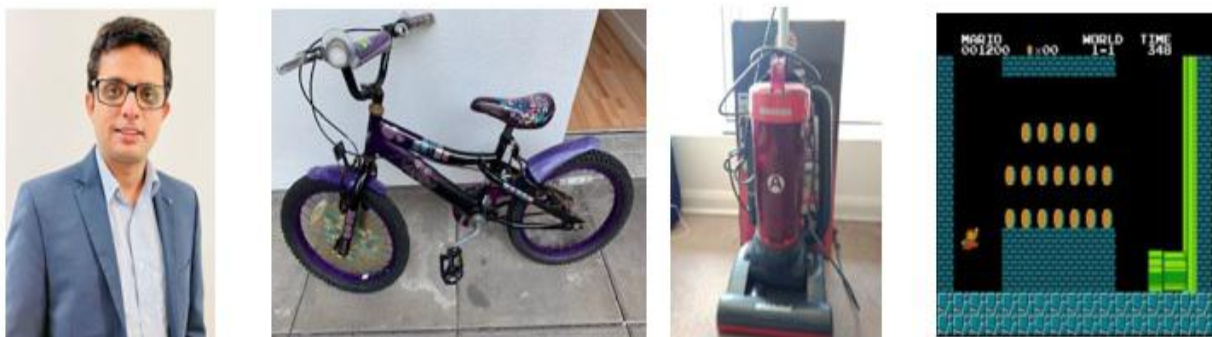
Image is a complex data point. Each image is made up of pixels, and each pixel can be made up of RGB (red, green, blue) values. And values for each of the red, green, blue can range from 0 to 255. The resulting dataset will be a very high-dimensional dataset.

Now, recall Principal Component Analysis (PCA) we studied in Chapter 3. PCA is a linear algorithm. And being a linear algorithm, its capability is limited to resolve non-linear and complex polynomial functions. Moreover, when a *high-dimensional* dataset has to represented in a low-dimensional space the algorithm should keep similar datapoints close to each other, which can be challenge in linear algorithms. PCA being a linear dimension reduction technique, it tries to separate the different data points as far away from each other as PCA is trying to maximize the variance between the data points. The resulting analysis is not robust and might not be best suited for further usage and visualization. Hence, we have non-linear algorithms like t-SNE to help.

Formally put, t-SNE is a *non-linear dimensionality reduction technique*

which is quite handy for high dimensional data. It is based on Stochastic Neighbor Embedding which was developed by Sam Roweis and Geoffrey Hinton. The t-distributed variant was proposed by Lauren van der Maaten. So, t-SNE is an improvement on the SNE algorithm.

At a high level, SNE measures the similarity between instances pairs in a high-dimensional space and in a low dimensional space. A good solution is where the difference between these similarity measures is the least and hence SNE then optimizes these similarity measure using a cost function.

We will examine the step-by-step process of t-SNE now. The process described below is a little heavy on mathematics.

1. Consider we have a high-dimensional space and we have some points in this high-dimensional space.
2. We will now measure the similarities between the various points. For a point x_i, we will then create a Gaussian distribution centered at that point. We have already studied Gaussian or normal distribution is the last chapters of the book. The Gaussian distribution is shown in Figure 6-10.

Figure 6-10 Gaussian or normal distribution which we have already studied earlier. Image has been taken from Wikipedia.
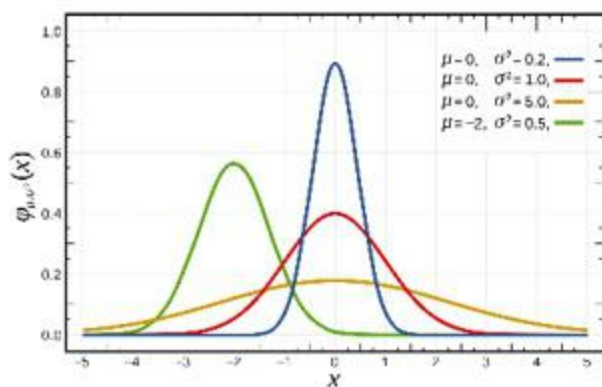
3. Now we will measure the density of points (let's say x_j) which fall under that Gaussian Distribution and then we renormalize them to get the respective conditional probabilities ($p_{j|i}$). For the points which are

nearby and hence similar, this conditional probability will be high and for the points which are far and dissimilar, the value of conditional probabilities ($p_{j|i}$) will be very small. These values of probabilities are the ones in high-dimensional space. For the curious ones, the mathematical formula for this conditional probability is Equation 6-3:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

(Equation 6-3)

where σ is the variance of the Gaussian Distribution centered at x_i. The mathematical proof is beyond the scope of this book.

4. Now we will measure one more set of probabilities in the low-dimensional space. For this set of measurements, we use *Cauchy Distribution*.

Cauchy Distribution

5. Cauchy distribution, belongs to the family of continuous probability distributions. Though there is a resemblance with the normal distribution, as we have represented in Figure 6-11, the Cauchy distribution has narrower peak and spreads out more slowly. It means that as compared to a normal distribution, the probability of obtaining values far from the peaks are higher. Sometimes, Cauchy distribution is also known as *Lorentz distribution.* It is interesting to note that Cauchy does not a well-defined mean but the median is the center of symmetry.

Figure 6-11 Comparison of Gaussian distribution vs Cauchy distribution. (Image source: Quora)
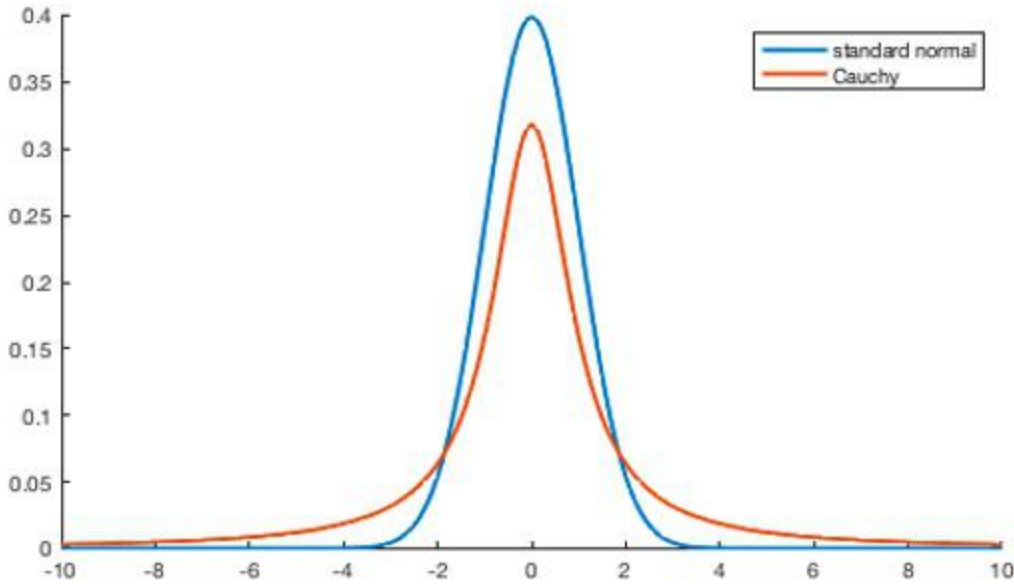
6. Consider we get y_i and y_j as the low-dimensional counterparts for the high-dimensional data points x_i and x_j. So, we can calculate the probability score like we did in the last step. Using Cauchy distribution, we can get second set of probabilities $q_{j|i}$ too. The mathematical formula is shown below in Equation 6-4.

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}.$$

(Equation 6-4)

7. So far, we have calculated two set of probabilities $(p_{j|i})$ and $(q_{j|i})$. In this step, we compare the two distributions and measure the difference between the two. In other words, while calculating $(p_{j|i})$ we measured the probability of similarity in a high-dimensional space whereas for $(q_{j|i})$ we did the same in a low-dimensional space. Ideally, if the mapping of the two spaces to be similar and for that there should be not be any difference between $(p_{j|i})$ and $(q_{j|i})$. So, the SNE algorithm tries to minimize the difference in the conditional probabilities $(p_{j|i})$ and $(q_{j|i})$.

8. The difference between the two probability distributions is done using Kullback-Liebler divergence or KL divergence, which we will explore here.

KL divergence

KL divergence or relative entropy is used to measure the difference between two probability distributions – usually one probability distribution is the data or the measured scores. The second probability distribution is an approximation or the prediction of the original probability distribution. For example, if the original probability distribution is X and the approximated one is Y. KL divergence can be used to measure the difference between X and Y probability distributions. In absolute terms if the value is 0 then it means that the two distributions are similar. The KL divergence is applicable for neurosciences, statistics and fluid mechanics.

9. To minimize the KL cost function, we use the gradient descent approach. We have already discussed the gradient descent approach in the section where we discussed MDS algorithm.

10. There is one more important point we should be aware while we work on t-SNE, an important hyperparameter called *perplexity*. Perplexity is a hyperparameter which allows us to control and optimize the number of close neighbors each of the data point has.

As per the official paper, a typical value for perplexity lies between 5 and 50.

11. There can be one additional nuance – the output of a t-SNE algorithm might never be same on successive runs. We have to optimize the values of the hyperparameters to receive the best output.

🎯 POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

1. Explain Cauchy distribution is your own words.

2. PCA is a non-linear algorithm. True or False.

3. KL divergence is used to measure the difference between two probability distributions. True or False

We will now proceed to the Python implementation of the algorithm.

6.3.1 Python implementation of t-SNE

We will use two datasets in this example. The first one is the already known IRIS dataset, which we have already used more than once in this book. The second dataset is quite an interesting one. It is MNIST dataset which is a database of handwritten digits. It is one of the most famous datasets used to train image processing solutions and generally is considered "Hello World" program for image detection solutions. An image representation is shown below in ().

Figure 6-12 MNIST dataset- it is a collection of handwritten images of digits.

Step 1: We will first import the necessary libraries. Note that we have imported MNIST dataset from keras library.

```
from sklearn.manifold import TSNE
from keras.datasets import mnist
from sklearn.datasets import load_iris
```

```
from numpy import reshape
import seaborn as sns
import pandas as pd
```

Step 2: First we will work with the IRIS dataset. We will load the IRIS dataset. The dataset comprises of two parts – one is the "data" and second is the respective label or "target" for it. It means that "data" is the description of the data and "target" is the type of IRIS. We are printing the features and the labels using a piece of code.

```
iris = load_iris()
iris_data = iris.data
iris_target = iris.target
iris.feature_names
iris.target_names
```

Step 3: The next step is invoking the tSNE algorithm. We are using the number of components = 2 and random_state =5 to reproduce the results. And then the algorithm is used to fit the data.

```
tsne = TSNE(n_components=2, verbose=1, random_state=5)
fitted_data = tsne.fit_transform(iris_data)
```

```
In [31]:    1  tsne = TSNE(n_components=2, verbose=1, random_state=5)
            2  fitted_data = tsne.fit_transform(iris_data)
            3

[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 150 samples in 0.000s...
[t-SNE] Computed neighbors for 150 samples in 0.003s...
[t-SNE] Computed conditional probabilities for sample 150 / 150
[t-SNE] Mean sigma: 0.509910
[t-SNE] KL divergence after 250 iterations with early exaggeration: 52.932037
[t-SNE] KL divergence after 1000 iterations: 0.123070
```

Step 4: We are now plotting the data. This step allows us to visualize the data fitted by the algorithm in the last step.

First, we will initiate an empty dataframe. We will add three columns one at a time. We will start with iris_target, followed by tSNE_first_component and tSNE_second_component. tSNE_first_component is the first column of the fitted_data dataframe and hence the index is 0. tSNE_second_component is

the second column of the fitted_data dataframe and hence the index is 1. Finally, we are representing the data in a scatterplot.

```
iris_df = pd.DataFrame()
iris_df["iris_target"] = iris_target
iris_df["tSNE_first_component"] = fitted_data[:,0]
iris_df["tSNE_second_component"] = fitted_data[:,1]

sns.scatterplot(x="tSNE_first_component", y="tSNE_second_componen
                palette=sns.color_palette("hls", 3),
                data=iris_df).set(title="Iris data tSNE projectio
```

Figure 6-13 tSNE projection of the IRIS dataset. Note how we are getting three separate clusters for the three classes we have in the dataset
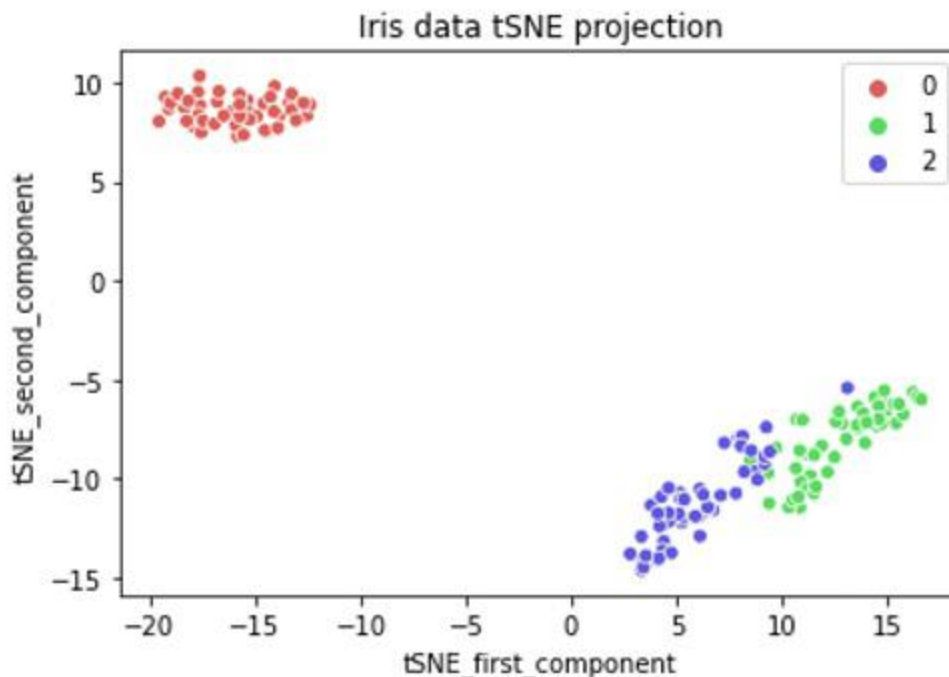
Now we will implement the algorithm for MNIST dataset.

Step 1: the libraries are already loaded in the last code example. Then we load the dataset. The dataset requires reshape which is done here

```
(digit, digit_label), (_ , _) = mnist.load_data()
digit = reshape(digit, [digit.shape[0], digit.shape[1]*digit.shap
Step 2: the subsequent steps are exactly same to the last example
tsne_MNIST = TSNE(n_components=2, verbose=1, random_state=5)
```

```
fitted_data = tsne_MNIST.fit_transform(digit)

mnist_df = pd.DataFrame()
mnist_df["digit_label"] = digit_label
mnist_df["tSNE_first_component"] = fitted_data[:,0]
mnist_df["tSNE_second_component"] = fitted_data[:,1]

sns.scatterplot(x="tSNE_first_component", y="tSNE_second_componen
                palette=sns.color_palette("hls", 10),
                data=mnist_df).set(title="MNIST data T-SNE projec
```

Figure 6-14 Output of tSNE for the 10 classes of digits represented in different colors.

MNIST data T-SNE projection
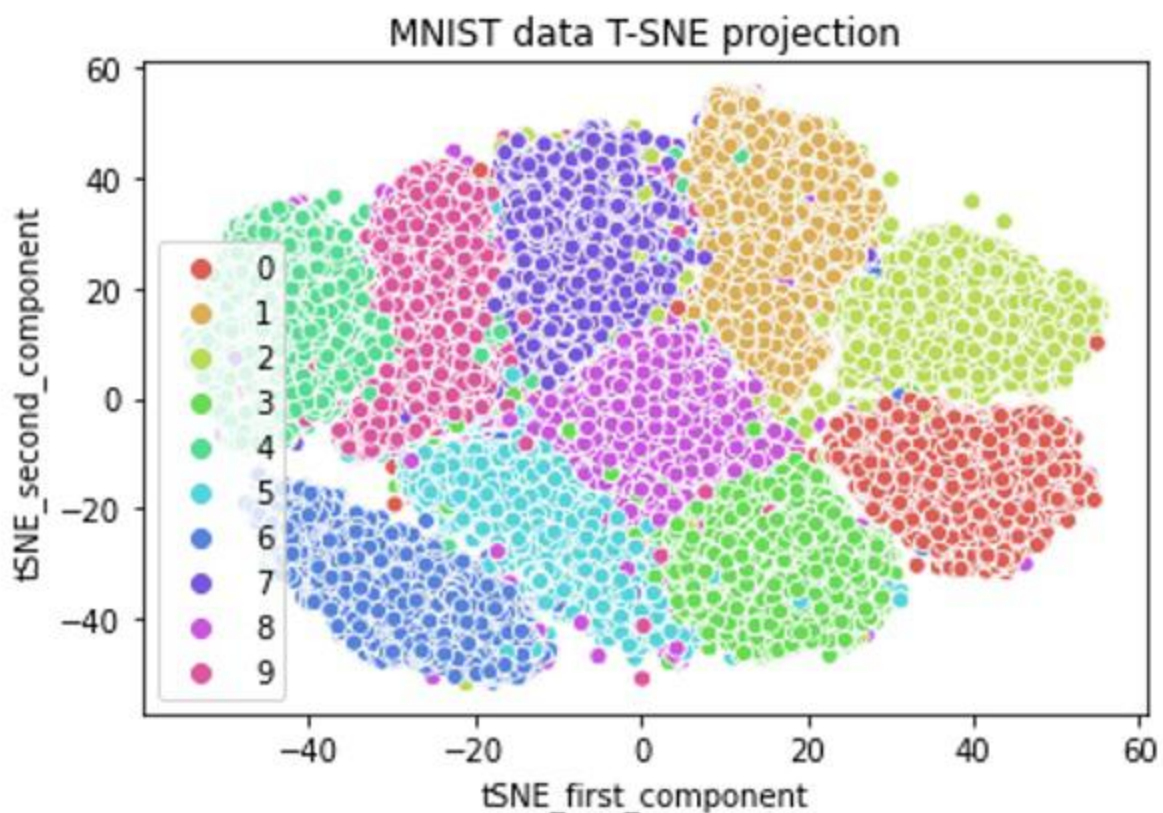
There are a few important points which you should keep in mind while running tSNE:

1. Run the algorithm with different values of hyperparameters before finalizing a solution.
2. Ideally, perplexity should be between 5 and 50 and for an optimized solution, the value of perplexity should be less than the number of

points.

3. tSNE guess the number of close neighbors for each of the point. And because of this reason, a dataset which is denser will require a much higher perplexity value.
4. Particularly, we should note that perplexity is the hyper parameter which balances the attention given to both the local and global aspects of the data.

tSNE is one of the widely popular algorithms. It is used for studying topology of an area, but a single tSNE cannot be used for making a final assessment. Instead, multiple tSNE plots should be created to make any final recommendation. Sometimes, there are complaints that tSNE is a black box algorithm. This might be true to a certain extent. What makes the adoption of tSNE harder is that it does not generate same results in successive iterations. Hence, you might find tSNE being recommended only for exploratory analysis.

This concludes our discussion on tSNE. We will now move to the case study.

6.4 Case study

Recall from chapter 3 where we explored a case study for telecom industry employing dimensionality. In this chapter, we will examine a small case study wherein tSNE or MDS can be utilized for dimensionality reduction.

Have you heard about hyperspectral images? As you know, we humans see the colors of visible light in mostly three bands – long wavelengths, medium ones and short wavelengths. The long wavelengths are perceived as red color, medium are green and short ones are perceived as blue color. Spectral imagining on the other hand, divides the spectrum into many greater numbers of bands and this technique can be extended beyond the visible ones and hence is of usage across biology, physics, geoscience, astronomy, agriculture and many more avenues.

Hyperspectral imaging collects and processes information from across the electromagnetic spectrum. It obtains the spectrum for each of the pixel in the image.

Figure 6-15 Hyperspectral image of "sugar end" potato strips shows invisible defects (Image source: Wikipedia)
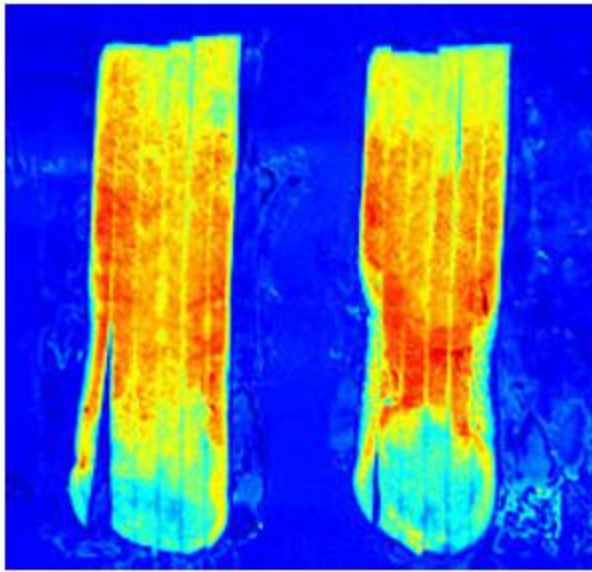
One such dataset can be the Pavia University Dataset. It is acquired by ROSIS sensor on Pavia, northern Italy. The details of the dataset are given below and the dataset can be downloaded from (http://www.ehu.eus/ccwintco/uploads/e/ee/PaviaU.mat http://www.ehu.eus/ccwintco/uploads/5/50/PaviaU_gt.mat)

In this dataset the spectral bands are 103, HIS size is 610*340 pixels and it contains 9 classes. Now, such a type of data can be used for crop analysis, mineral examining and explorations etc. Since this data contains information about the geological patterns, it is quite useful for scientific purpose. Before developing any image recognition solutions, we have to reduce the number of dimensions for this dataset. Moreover, the computation cost will be much higher if we have a large number of dimensions. Hence, it is obvious to have lesser number of representative number of dimensions. We are showing a few example bands of below. You are advised to download the dataset (which is also checked-in at the git Hub repo) and use the various dimensionality reduction techniques on the dataset to reduce the number of dimensions.

Figure 6-16 Example of bands in the dataset. These are only random examples, you are advised to load the dataset and run dimensionality reduction algorithms.

Band - 25   Band - 9   Band - 53

Band - 29   Band - 63   Band - 14
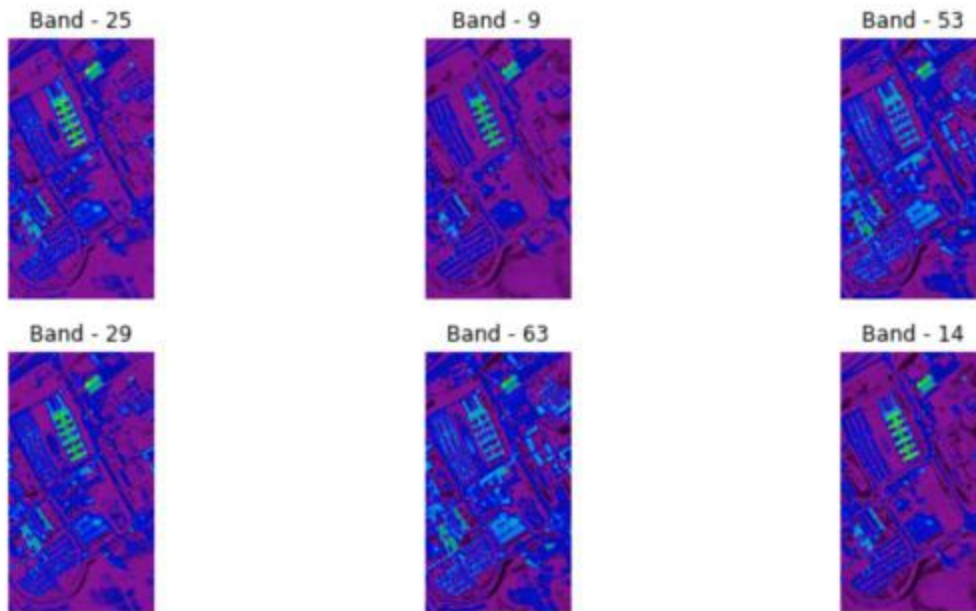
There can be many other image datasets and complex business problems where tSNE and MDS can be of pragmatic usage. Some of such datasets have been listed in the next steps.

6.5 Summary

Dimensionality reduction is quite an interesting and useful solution. It makes the machine learning less expensive and time consuming. Imagine that you have a dataset with thousands of attributes or features. You do not know the data very well; the business understanding is quite less and at the same time you have to find the patterns in the dataset. You are not even sure that if these variables are all relevant or just random noise. At such a moment, when we have to quickly reduce the number of dimensions in the dataset, make it less complex to crack and reduce the time – dimensionality reduction is the solution.

We covered dimensionality reduction techniques earlier in the book. This chapter covers two advanced techniques – tSNE and MDS. Both of these techniques should not be considered a substitute to the other easier techniques we discussed. Rather, they are two be used if we are not getting meaningful results. It is always advised to use PCA first, then try tSNE or MDS.

We are increasing the complexity in the book. This chapter started with images – we have only wet our toe though. In the next chapter, we are dealing with text data, perhaps you will find it very interesting and useful.

Practical next steps and suggested readings

1. Use the vehicles dataset used in the Chapter2 for clustering and implement MDS on it. Compare the performance on clustering before and after implementing MDS.
2. Get the datasets used in Chapter 2 for Python examples and use them for implementing MDS.
3. For MDS, you can refer to the following research papers:
4. Dimensionality reduction: a comparative review by Lauren van der Maaten, Eric Postma and H. Japp Van Den Herik
    https://www.researchgate.net/publication/228657549_Dimensionality_R
5. Multidimensional scaling -based data dimension reduction method for application in short term traffic flow prediction for urban road network by Satish V. Ukkusuri and Jian Lu
    https://www.hindawi.com/journals/jat/2018/3876841/
6. Get tSNE research papers from the links below and study them
7. Visualizing data using t-SNE by Laurens van der Maaten and Geoffrey Hinton
    https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08
8. The art of using t-SNE for single cell transcriptomics
    https://www.nature.com/articles/s41467-019-13056-x
9. There is one more paper which might be of interest- Performance evaluation of t-SNE and MDS dimensionality reduction techniques with KNN, SNN and SVM classifiers https://arxiv.org/pdf/2007.13487.pdf

7 Unsupervised learning for text data

"Everybody smiles in the same language – George Carlin"

Our world has so many languages. These languages are the most common medium of communication to express our thoughts and emotions to each other. This ability to express our thoughts in words is unique to humans. These words are a source of information to us. These words can be written into text. In this chapter, we are going to explore the analysis we can do on text data. Text data falls under unstructured data and carries a lot of useful information and hence is a useful source of insights for the business. We use natural language processing or NLP to analyse the text data.

At the same time, to analyse text data, we have to make the data analysis ready. Or in very simple terms, since our algorithms and processors can only understand numbers, we have to represent the text data in numbers or *vectors*. We are exploring all such steps in this chapter. Text data holds the key to quite a few important use cases like sentiment analysis, document categorization, language translation etc. to name a few. We will cover the use cases using a case study and develop Python solution on the same.

The chapter starts with defining text data, sources of text data and various use cases of text data. We will then move to the steps and processes to clean and handle the text data. We will cover the concepts of NLP, mathematical foundation and methods to represent text data into vectors. We will create Python codes for the use cases. And at the end, we are sharing case study on text data. In this book, we are providing that level of mathematical support without going in too much depth – an optimal mix of practical world and mathematical concepts.

In this seventh chapter of the book, we are going to cover the following topics:

1. Text data and various use cases of Text data analysis
2. Challenges we face with text data
3. Pre-processing of text data and data cleaning
4. Methods to represent text data in vectors
5. Sentiment analysis using Python – a case study
6. Text clustering using Python

Welcome to the seventh chapter and all the very best!

7.1 Technical toolkit

We will continue to use the same version of Python and Jupyter notebook as we have used so far. The codes and datasets used in this chapter have been checked-in at this location.

You would need to install a few Python libraries in this chapter which are – XXXX. Along with this we will need numpy and pandas. Using libraries, we can implement the algorithms very quickly. Otherwise, coding these algorithms is quite a time-consuming and painstaking task.

Here we are dealing with text data, perhaps you will find it very interesting and useful.

Let's get started with Chapter 7 of the book!

7.2 Text data is everywhere

Recall in the very first chapter of the book, we explored structured and unstructured datasets. Unstructured data can be text, audio, image or a video. The examples of unstructured data and their respective sources are given in (Figure 7-1) below, where we explain the primary types of unstructured data: text, images, audio and video along with their examples. The focus for this chapter is on text data.

Figure 7-1 Unstructured data can be text, images, audio, video. We are dealing with text data in this chapter

Unstructured Data

Text Data	Facebook reviews, tweets, customer complaints, product reviews
Images Data	Product Images, Objects
Audio Data	Call center recordings, radio ads
Video Data	YouTube videos, product videos, video ads, product shoots
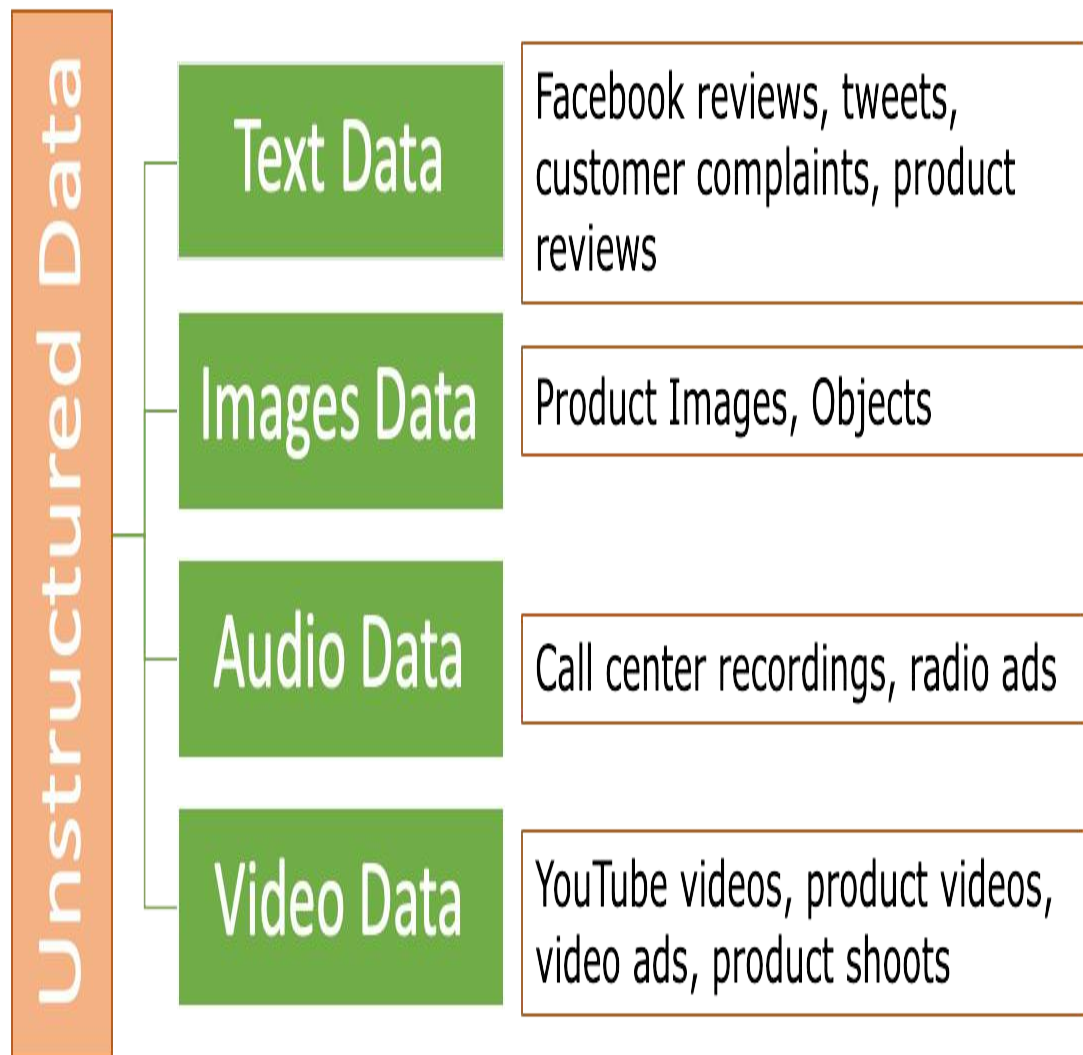
Language is indeed a gift to the mankind. We indulge in speaking, messaging, writing, listening to convey our thoughts. And this is done using text data which is generated using blogs and social media posts, tweets, comments, stories, reviews, chats and comments to name a few. Text data is generally much more direct, and emotionally expressive. It is imperative that business unlocks the potential of text data and derives insights from it. We can understand our customers better, explore the business processes and gauge the quality of services offered. We generate text data is the form of news, Facebook comments, tweets, Instagram posts, customer reviews, feedback, blogs, articles, literature, stories etc. This data represents a wide range of emotions and expressions.

Have you even reviewed a product or a service on Amazon? You award stars to a product; at the same time, you can also input free text. Go to Amazon and look at some of the reviews. You might find some reviews have good amount of text as the feedback. This text is useful for the product/service providers to enhance their offerings. Also, you might have participated in a few surveys which ask you to share your feedback. Moreover, with the advent of Alexa, Siri, Cortona etc. the voice command is acting as an interface between humans and machines – which is again a rich source of data. Even the customer calls we make to a call centre can be source of text data. These calls can be recorded and using speech-to-text conversion, we can generate huge amount of text data. Massive dataset, right!

We are discussing some of the important use cases for text data in the following section.

7.3 Use cases of text data

Text data is indeed super useful. It is really a rich source of insights for the business. There are some of the use cases which are of much interest, which we are listing below. The list is not exhaustive. At the same time, not all the use cases given implement unsupervised learning. Some of the use cases require supervised learning too. Nevertheless, for your knowledge we are sharing both types of use cases – based on supervised learning and unsupervised learning.

1. **Sentiment analysis:** You might have participated in surveys or given your feedback of products/surveys. These survey generate tons of text data for us. That text data can be analyzed and we can determine whether the sentiment in the review is positive or negative. In simple words, sentiment analysis is gauging what is the positiveness or negativity in the text data. And hence, what is the sentiment about a product or service in the minds of the customers. We can use both supervised and unsupervised learning for sentiment analysis.
2. **News categorization or document categorization:** Look at the Google News webpage, you will find that each news item has been categorized to sports, politics, science, business or any other category. Incoming news will be classified based on the content of the news which is the

actual text. Similarly, imagine we have some documents with us and we might want to segregate them based on their categories of based on the domain of study. For example, medical, economics, history, arts, physics etc. Such a use case will save a lot of time and man power indeed.

3. **Language Translation:** Translation of text from one language to another is a very interesting use case. Using natural language processing we can translate between languages. Language translation is very tricky as different languages have different grammatical rules. Generally, deep learning based solutions are the best fit for language translation.

4. **Spam filtering:** email spam filter can be composed using NLP and supervised machine learning. A supervised learning algorithm can analyze incoming mail parameters and can give a prediction if that email belongs to a spam folder or not. The prediction can be based on the various parameters like sender email-id, subject line, body of the mail, attachments, time of mail etc. Generally, supervised learning algorithms are used here.

5. **Part of speech tagging** or POS tagging is one of the popular use cases. It means that we are able to distinguish the noun, adjectives, verbs, adverbs etc. in a sentence. **Named-entity recognition** or NER is also one of the famous applications of NLP. It involves identifying a person, place, organization, time, number in a sentence. For example, John lives in London and works for Google. NER can generate understanding like [John]$_{Person}$ lives in [London]$_{Location}$ and works for [Google]$_{organization}$.

6. Sentence generation, captioning the images, speech-to-text or text-to-speech tasks, handwriting recognition are a few other significant and popular use cases.

The use cases listed above are not exhaustive. There are tons of other use cases which can be implemented using NLP. NLP is a very popular research field too. We are sharing some significant papers at the end of the chapter for your perusal.

Whilst, text data is of much importance, at the same time is quite a difficult dataset to analyze. To be noted is, our computers and processors understand only numbers. So, the text still needs to be represented as numbers so that we can perform mathematical and statistical calculations on them. But before diving into the preparation of text data, we will cover some of the challenges

we face while working on text dataset.

7.4 Challenges with text data

Text is perhaps the most difficult data to work with. There are a large number of permutations to express the same thought. For example, if I may ask, "Hey buddy, what is your age?" and "Hey buddy, may I know how old are you?" mean the same, right! The answer to both the questions is same, and it is quite easy for humans to decipher. But can be an equally daunting task for a machine.

The most common challenges we face are:

1. Text data can be complex to handle. There can be a lot of junk characters like $^%*& present in the text.
2. With the advent of modern communications, we have started to use short forms of words like "u" can be used for "you", "brb" for "be right back" and so on.
3. Language is changing, unbounding and ever evolving. It changes every day and new words are added to the language.

If you do a simple Google search, you will find that quite a few words are added to the dictionary each year.

4. The world has close to 6500 languages, and each and every one carries their uniqueness. Each and every one complete our world. For example, Arabic, Chinese, English, French, German, Hindi , Italian, Japanese, Spanish etc. Each language follows its own rules and grammar which are unique in usage and pattern. Even the writing can be different - some are written left to right; some might be right to left or may be vertically! The same emotion, might take lesser or a greater number of words in different languages.
5. The meaning of a word is dependent on the context. A word can be an adjective and can be a noun too depending on the context. Look at these examples below:
6. "This book is a must read" and "Please book a room for me".
7. "Tommy" can be a name but when used as "Tommy Hilfiger" its usage

is completely changed.

8. "Apple" is a fruit while "Apple" is a company producing Macintosh, iPhones etc.
9. "April" is a month and can be a name too.
10. Look at one more example - "Mark travelled from the UK to France and is working with John over there. He misses his friends". The humans can easily understand that "he" in the second sentence is Mark and not John, which might not be that simple for a machine.
11. There can be many synonyms for the same word, like "good" can be replaced by positive, wonderful, superb, exceptional in different scenarios. Or, words like "studying", "studying", "studies", "studies" are related to the same root word "study".
12. And the size of text data can be daunting too. Managing a text dataset, storing it, cleaning it and refreshing it is a herculean task in itself.

Like any other machine learning project, text analytics follow the principles of machine learning albeit the precise process is slightly different. Recall in chapter 1, we examined the process of a machine learning project as shown in Figure 7-2. You are advised to refresh the process from Chapter 1.

Figure 7-2 Overall steps in data science project are same for text data too. The pre-processing of text data is very different from the structured dataset.
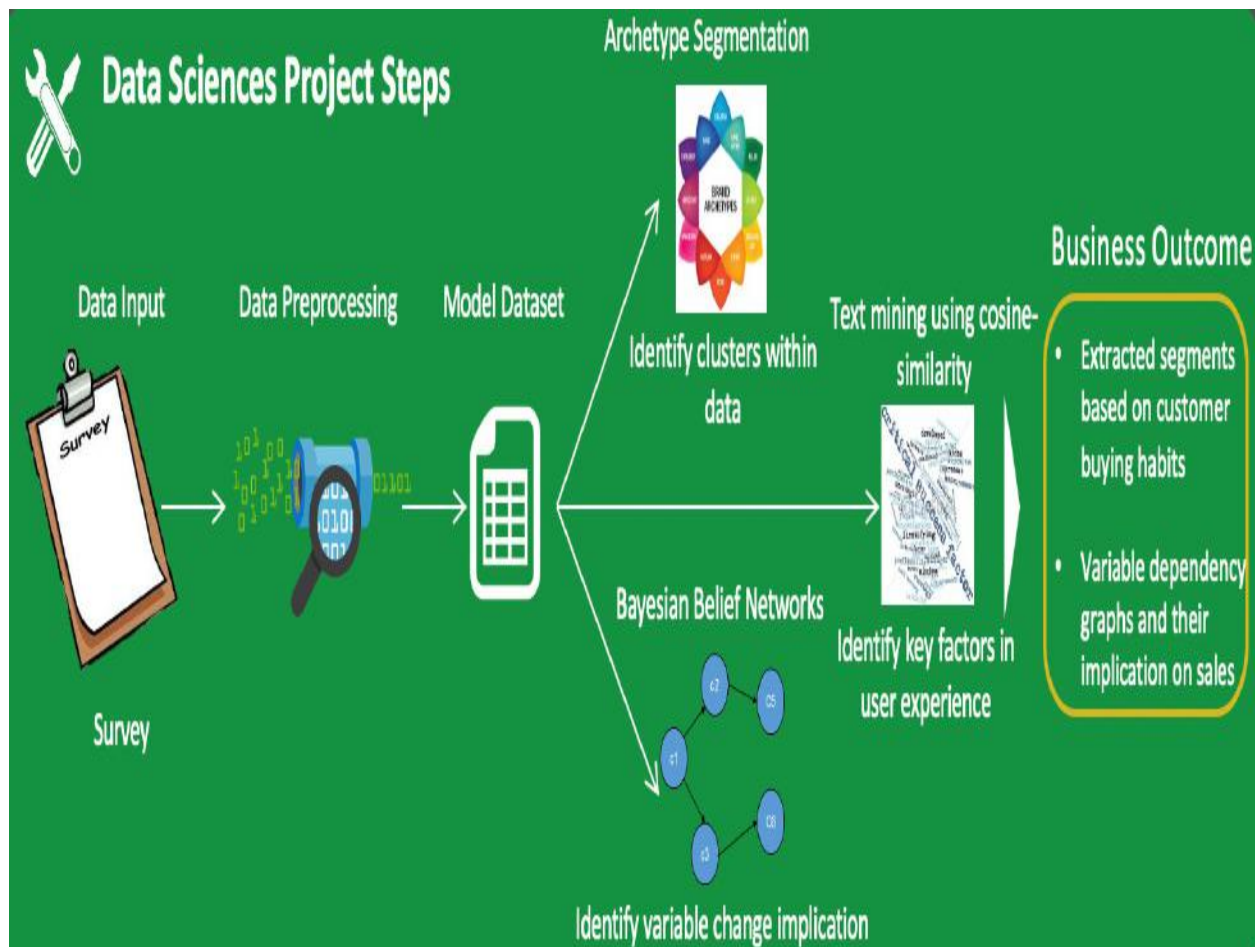
Defining the business problem, data collection and monitoring etc. remain the same. The major difference is in processing of the text, which involves data cleaning, creation of features, representation of text data etc. We are covering it now.

☐

POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

(1)   Note the three most impactful use cases for the text data.

(2)   Why is working on text data so tedious?

7.5 Preprocessing of the text data

Text data, like any other data source can be messy and noisy. We clean some of it in the data discovery phase and a lot of it in the pre-processing phase. At the same time, we have to extract the features from our dataset. This cleaning process is sometimes and can be implemented on most of the text datasets. Some text datasets might require a customized approach. We will start with cleaning the raw text data.

7.5.1 Data cleaning

There is no second thought about the importance of data quality. The cleaner the text data is, the better the analysis will be. At the same time, the pre-processing is not a straight-forward task. It is complex and time-consuming task.

Text data is to be cleaned as it contains a lot of junk characters, irrelevant words, noise and punctuations, URLs etc. The primary ways of cleaning the text data are:

1. **Stop words removal:** out of all the words that are used in any language, there are some words which are most common. Stop words are the most common words in a vocabulary which carry less importance than the key words. For example, "is", "an", "the", "a", "be", "has", "had", "it" etc. Once we remove the stop words from the text, the dimensions of the data are reduced and hence complexity of the solution is reduced.
At the same time, it is imperative that we understand the context very well while removing the stop words. For example, if we ask a question "is it raining?". Then the answer "it is" is a complete answer in itself. To remove the stop words, we can define a customized list of stop words and remove them. Else there are standard libraries are to remove the stop words.

When we are working with solutions where contextual information is important, we do not remove stop words.

2. **Frequency based removal of words:** Sometimes, you might wish to remove the words which are most common in your text or very unique. The process is to get the frequency of the words in the text and then set a

threshold of frequency. We can remove the most common ones. Or maybe you might wish to remove the ones which have occurred only once/twice in the entire dataset. Based on the requirement, you will decide.

3. **Library based cleaning** is done when we wish to clean the data using pre-defined and customized library. We can create a repository of words which we do not want in our text and can iteratively remove from the text data. This approach allows us flexibility to implement the cleaning of our own choice.

4. **Junk or unwanted characters:** A text data particularly tweets, comments etc. might contain a lot of URLs, hashtags, numbers, punctuations, social media mentions, special characters etc. We might need to clean them from the text. At the same time, we have to be careful as some words which are not important for one domain might be quite required for a different domain. If data has been scraped from websites or HTML/XML sources, we need to get rid of all the HTML entities, punctuations, non-alphabets and so on.

Always keep business context in mind while cleaning the text data.

As we know that a lot of new type of expressions have entered the language. For example, lol, hahahaha, brb, rofl etc. These expressions have to be converted to their original meanings. Even emojis like :-), ;-) etc. have to be converted to their original meanings.

5. **Data encoding:** There are a number of data encodings available like ISO/IEC, UTF-8 etc. Generally, UTF-8 is the most popular one. But it is not a hard and fast rule to always use UTF-8 only.

6. **Lexicon normalization:** Depending on the context and usage, the same word might get represented in different manners. During lexicon normalization we clean such ambiguities. The basic idea is to reduce the word to its root form. Hence, words which are derived from each other can be mapped to the central word provided they have the same core meaning.
   Look at Figure 7-2 wherein we have shown that the same word "eat", has been used in various forms. The root word is "eat" but these different forms are so many different representations for eat.

Figure 7-3 Ate, eaten, eats, eating all have the same root word – eat. Stemming and lemmatization can be used to get the root word.
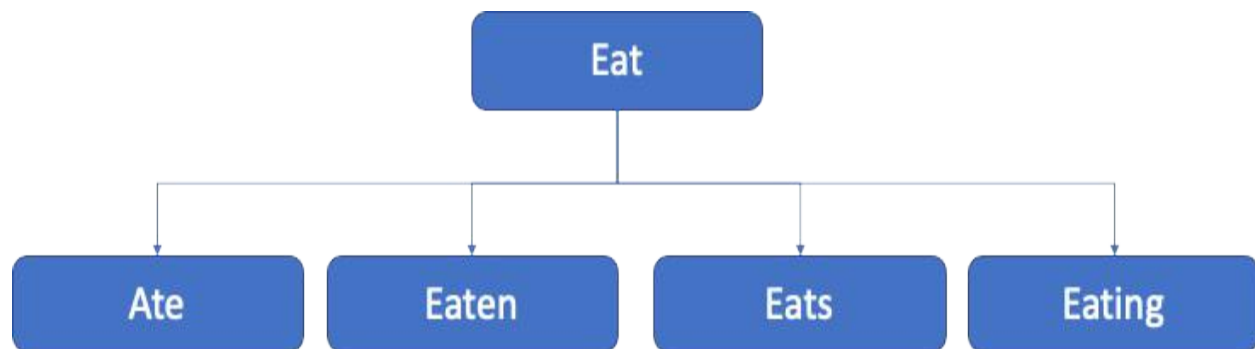
We wish to map all of these words like eating, eaten etc. to their central word "eat", as they have the same core meaning. There are two primary methods to work on this:

  a. **Stemming**: *Stemming* is a basic rule-based approach of mapping a word to its core word. It removes "es", "ing", "ly", "ed" etc. from the end of the word. For example, studies will become studi and studying will become study. As visible being a rule-based approach, the output spellings might not always be accurate.
  b. **Lemmatization**: is an organized approach which reduces words to their dictionary form. *Lemma* of a word is its dictionary or canonical form. For example, eats, eating, eaten etc. all have the same root word eat. Lemmatization provides better results than stemming but it takes more time then stemming.

These are only some of the methods to clean the text data. These techniques will help to a large extent. But still business acumen is required to further make sense to the dataset. We will clean the text data using these approaches by developing Python solution.

Once the data is cleaned, we have to start representation of data so that it can be processed by machine learning algorithms – which is our next topic.

7.5.2 Extracting features from the text dataset

Text data, like any other data source can be messy and noisy. We explored

the concepts and techniques to clean it in the last section. Now we have cleaned the data and it is ready to be used. The next step is to represent this data in a format which can be understood by our algorithms. As we know that our algorithms can only understand numbers. Text data in its purest form cannot be understood by algorithms. So, everything needs to be converted to numbers.

A very simple technique can be to simply perform *one-hot encoding* on our words and represent them in a matrix.

We have covered one hot encoding in the previous chapters of the book

If we describe the steps, the words can be first converted to lowercase and then sorted in an alphabetical order. And then a numeric label can be assigned to them. And finally, words are converted to binary vectors. Let us understand using an example.

For example, the text is "It is raining heavily". We will use the steps below:

  a. Lowercase the words so the output will be "it is raining heavily"
  b. We will now arrange them in alphabetical order. The result is – heavily, is, it, raining.
  c. We can now assign place values to each word as heavily:0, is:1, it:2, raining:3.
  d. Finally, we can transform them to binary vectors as shown below

```
[[0. 0. 1. 0.] #it
[0. 1. 0. 0.] #is
[0. 0. 0. 1.] #raining
[1. 0. 0. 0.]] #heavily
```

Though this approach is quite intuitive and simple to comprehend, it is pragmatically not possible due to the massive size of the corpus and the vocabulary.

Corpus refers to a collection of texts. It is Latin for body. It can be a body of written words or spoken words, which will be used to perform a linguistic analysis.

Moreover, handling massive data size with so many dimensions will be computationally very expensive. The resulting matrix thus created will be very sparse too. Hence, we look at other means and ways to represent our text data.

There are better alternatives available to one-hot encoding. These techniques focus on the frequency of the word or the context in which the word is being used. This scientific method of text representation is much more accurate, robust and explanatory. It generates better results too. There are multiple such techniques like tf-idf, bag-of-words approach etc. We are discussing a few of these techniques in the next sections. But we will examine an important concept of tokenization first!

Tokenization

Tokenization is simply breaking a text or a set of text into individual tokens. It is the building block of NLP. Look at the example in Figure 7-3, where we have created individual tokens for each of the word of the sentence. Tokenization is an important step as it allows us to assign unique identifiers or tokens to each of the words. Once we have allocated each word a specific token, the analysis become less complex.

Figure 7-3 Tokenization can be used to break a sentence into different tokens of words.
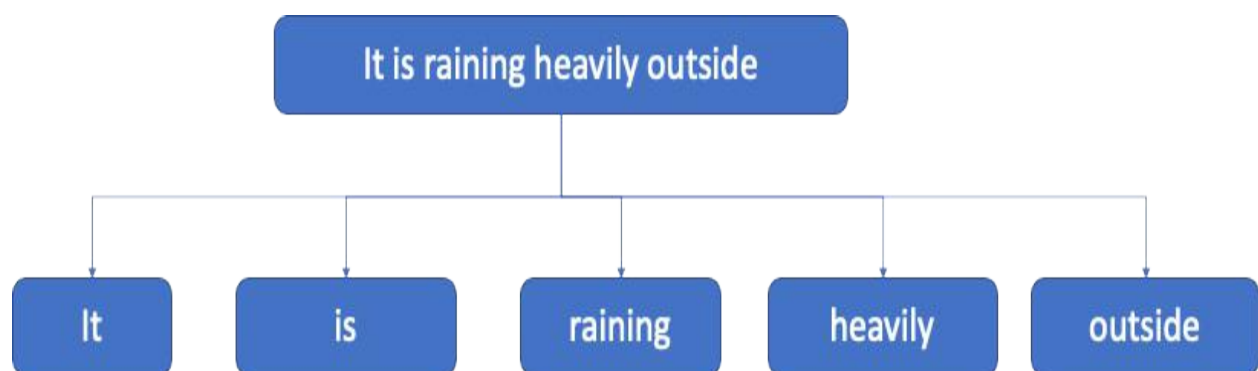
Tokens are usually used on individual words, but it is not always necessary. We are allowed to tokenize a word or the sub-words or characters in a word. In the case of sub-words, the same sentence can have sub-word tokens as rain-ing.

If we wish to perform tokenization at a character level, it can be r-a-i-n-i-n-g. In fact, in the one-hot encoding approach discussed in the last section as a first step, tokenization was done on the words.

Tokenization is the building blocks for Natural Language Processing solutions.

Once we have obtained the tokens, then the tokens can be used to prepare a vocabulary. Vocabulary is the set of unique tokens in the corpus.

There are multiple libraries for tokenization. *Regexp* tokenization uses the given patterns arguments to match the tokens or separators between the tokens. *Whitespace* tokenization uses by treating any sequence of whitespace characters as a separator. Then we have *blankline* which use sequence of blank lines as a separator. And *wordpunct* tokenizes by matching sequence of alphabetic characters and sequence of non-alphabetic and non-whitespace characters. We will perform tokenization when we create Python solutions for our text data.

Now, we will explore more methods to represent text data. The first such method is Bag of Words.

Bag of words approach

As the name suggests, all the words in the corpus are used. In bag of words approach, or BOW, the text data is tokenized for each word in the corpus and then the respective frequency of each token is calculated. During this process, we disregard the grammar, or the order or the context of the word. We simply focus on the simplicity. Hence, we will represent each text (sentence or a document) as a *bag of its own words*.

In the BOW approach for the entire document, we define the vocabulary of the corpus as all of the unique words present in the corpus. Please note we use all the unique words in the corpus. If we want, we can also set a threshold i.e., the upper and lower limit for the frequency of the words to be selected. Once we have got the unique words, then each of the sentence can be represented by a vector of the same dimension as of the base vocabulary

vector. This vector representation contains the frequency of each word of the sentence in the vocabulary. It might sound complicated to understand, but it is actually a straightforward approach.

Let us understand this approach with an example. Let's say that we have two sentences – It is raining heavily and We should eat fruits.

To represent these two sentences, we will calculate the frequency of each of the word in these sentences as shown in Figure 7-4.

Figure 7-4 The frequency of each word has been calculated. In this example, we have two sentences.

Words	Freq
It	1
is	1
raining	1
heavily	1

Words	Freq
We	1
should	1
eat	1
fruits	1

It is raining heavily

We should eat fruits

Now, if we assume that only these two words represent the entire vocabulary, we can represent the first sentence as shown in Figure 7-5. Note that the table contains all the words, but the words which are not present in the sentence have received value as 0.

Figure 7-5 The first sentence if represented for all the words in the vocabulary, we are assuming that in the vocabulary only two sentences are present

Words	Freq
eat	0
fruits	0
heavily	1
is	1
it	1
raining	1
should	0
we	0

It is raining heavily

In this example, we examined how BOW approach has been used to represent a sentence as a vector. But BOW approach has not considered the order of the words or the context. It focuses only on the frequency of the word. Hence, it is a very fast approach to represent the data and computationally less expensive as compared to its peers. Since it is frequency based, it is commonly used for document classifications.

But, due to its pure frequency-based calculation and representation, the solution accuracy can take a hit. In language, the context of the word plays a significant role. As we have seen earlier, apple is both a fruit as well as a well-known brand and organization. And that is why we have other advanced methods which consider more parameters than frequency alone. One of such methods is tf-idf or term frequency-inverse document frequency, which we are studying next.

POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

(1)   Explain tokenization in simple language as if you are explaining to a person who does not know NLP.

(2)   Bag of words approach the context of the words and not frequency alone. True or False.

(3)   Lemmatization is less rigorous approach then stemming. True or False.

tf-idf (Term frequency and inverse document frequency)

We studied Bag of words approach in the last section. In the BOW approach, we gave importance to the frequency of a word only. The idea is that the words which have higher frequency might not offer meaningful information as compared to words which are rare but carry more importance. For example, if we have a collection of medical documents, and we wish to compare two words "disease" and "diabetes". Since the corpus consists of medical documents, the word disease is bound to be more frequent whilst the word "diabetes" will be less frequent but more important to identify the documents which deal with diabetes. The approach tf-idf allow us to resolve this issue and extract information on the more important words.

In term-frequency and inverse-document-frequency (tf-idf), we consider the relative importance of the word. TF-idf means term frequency and idf means inverse document frequency. We can define tf-idf as:

a. **Term frequency (t** is the count of a term in the entire document. For example, the count of the word "a" in the document "D".
b. **Inverse document frequency (id** is the log of the ratio of total documents (N) in the entire corpus and number of documents(df) which contain the word "a".

So, the tf-idf formula will give us the relative importance of a word in the entire corpus. The mathematical formula is the multiplication of tf and idf and is given by

$w_{i,j} = tf_{i,j} * \log (N/df_i)$ (Equation 7-1)

where N: total number of documents in the corpus

$tf_{i,j}$ is the frequency of the word in the document

df_i is the number of documents in the corpus which contain that word.

The concept might sound complex to comprehend. Let's understand this with an example.

Consider we have a collection of 1 million sports journals. These sports journals contain have many articles of various lengths. We also assume that all the articles are in English language only. So, let's say, in these documents, we want to calculate tf-idf value for the word "ground" and "backhand".

Let's assume that there is a document of 100 words having "ground" word five times and backhand only twice. So tf for ground is 5/100 = 0.05 and for backhand is 2/100 = 0.02.

We understand that the word "ground" is quite a common word in sports, while the word "backhand" will have lesser number of usage. Now, we assume that "ground" appears in 100,000 documents out of 1 million documents while "backhand" appears only in 10. So, idf for "ground" is log (1,000,000/100,000) = log (10) = 1. For "backhand" it will be log (1,000,000/10) = log (100,000) = 5.

To get the final values for "ground" we will multiply tf and idf = 0.05 x 1 = 0.05.

To get the final values for "backhand" we will multiply tf and idf = 0.02 x 5 = 0.1.

We can observe that the relative importance of "backhand" is more than the relative importance of the word "ground". This is the advantage of tf-idf over frequency-based BOW approach. But tf-idf takes more time to compute as compared to BOW since all the tf and idf have to be calculated. Nevertheless, tf-idf offer a better and more mature solution as compared to BOW approach.

We will now cover language models in the next section.

Language models

So far, we have studied the bag of words approach and tf-idf. Now we are focusing on language models.

Language models assign probabilities to the sequence of words. N-grams are the simplest in language models. We know that to analyze the text data they have to be converted to feature vectors. N-gram models create the feature vectors so that text can be represented in a format which can be analyzed further.

n-gram is a probabilistic language model. In a n-gram model we calculate the probability of the Nth word given the sequence of (N-1) words. If we go more specific, an n-gram model will predict the next word x_i based on the words $x_{i-(n-1)}$, $x_{i-(n-2)}\ldots x_{i-1}$. If we wish to use the probability terms, we can represent as conditional probability of x_i given the previous words which can be represented as $P(x_i \mid x_{i-(n-1)}, x_{i-(n-2)}\ldots x_{i-1})$. The probability is calculated by using the relative frequency of the sequence occurring in the text corpus.

If the items are words, n-grams may be referred to as *shingles*.

Let's study this using an example.

Consider we have a sentence It is raining heavily. We have shown the respective representations using different values of n. You should note that how the sequence of words and their respective combinations is getting changed for different values of n. If we wish to use n=1 or a single word to make a prediction, the representation will be as shown in Figure 7-6. Note that each word is used separately here. They are referred to as *unigrams*.

If we wish to use n=2, the number of words now used will become two. They are referred to as *bigrams* and the process will continue.

Figure 7-6 Unigrams, bigrams, trigrams can be used to represent the same sentence. The concept can be extended to n-grams too.
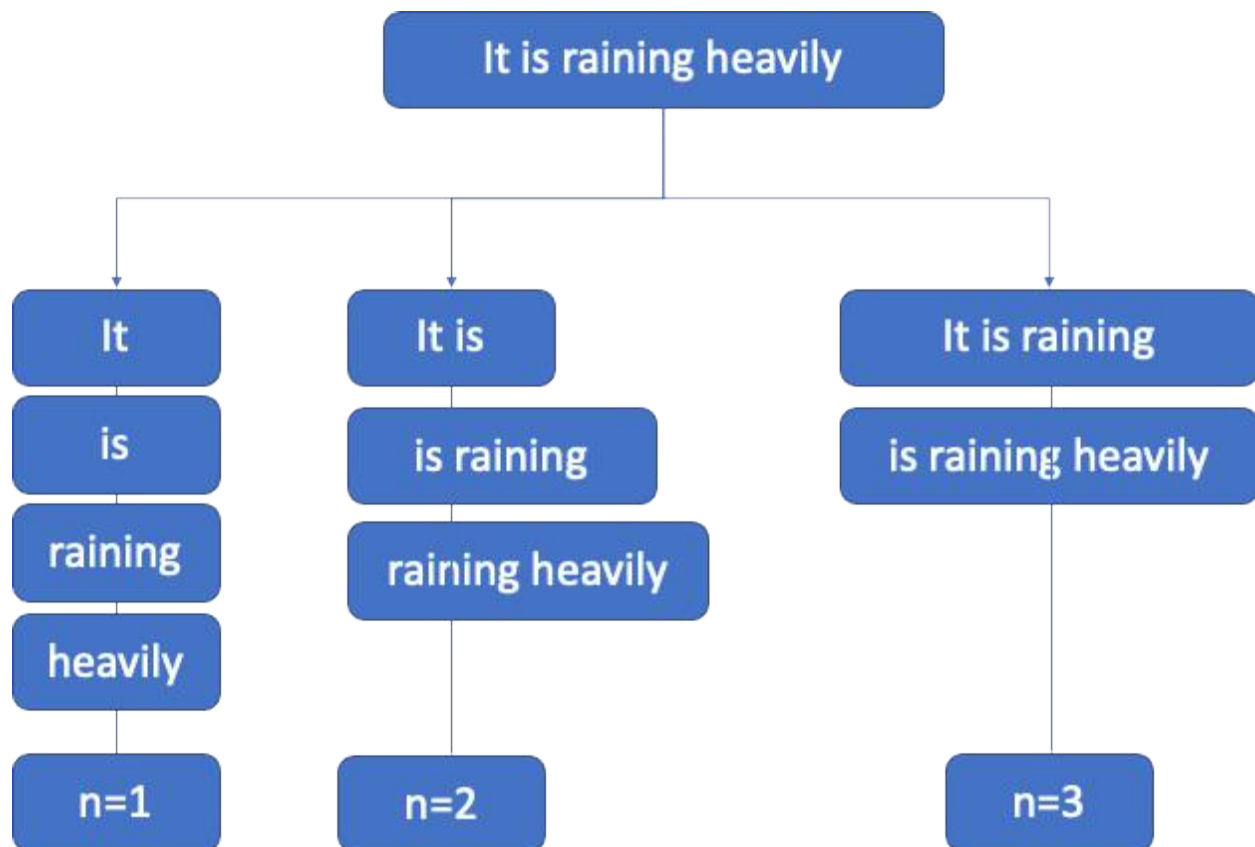
Hence, if we are having a unigram, it is a sequence of one word, for two words it is bi-gram, for three words it is tri-gram and so on. So, a tri-gram model will approximate the probability of a word given all the previous words by using the conditional probability of only the preceding two words. Whereas a bi-gram will do the same by considering only the preceding word. This is a very strong assumption indeed that the probability of a word will depend only on the preceding word and is referred to as *Markov* assumption. Generally, N > 1 is considered to be much more informative than unigrams. But obviously the computation time will increase too.

n-gram approach is very sensitive to the choice of n. It also depends significantly on the training corpus which have been used, which makes the probabilities heavily dependent on the training corpus. So, if an unknown word is encountered, it will be difficult for the model to work on that new word.

We will create a Python example now. We will show a few examples of text cleaning using Python.

Text cleaning using Python

We will clean the text data using Python now. There are a few libraries which you may need to install. We will show a number of small code snippets. You are advised to use them as per the requirement. We are also pasting the respective screenshot of the code snippets and their results.

Code 1: Remove the blank spaces in the text. We import the library `re`, it is called `Regex` expression. The text is It is raining outside with a lot of blank spaces in between.

```
import re
doc = "It is      raining        outside"
new_doc = re.sub("\s+"," ", doc)
print(new_doc)
```

```
1  import re
2  doc = "It is      raining        outside"
3  new_doc = re.sub("\s+"," ", doc)
4  print(new_doc)
```

```
It is raining outside
```

Code 2: Now we will remove the punctuations in the text data.

```
text_d = "Hey!!! How are you doing? And how is your health! Bye,
re.sub("[^-9A-Za-z ]", "" , text_d)
```

```
1  text_d = "Hey!!! How are you doing? And how is your health! Bye, take care."
2  re.sub("[^-9A-Za-z ]", "" , text_d)
```

'Hey How are you doing And how is your health Bye take care'

Code 3: This is one more method to remove the punctuation.

```
import string
text_d = "Hey!!! How are you doing? And how is your health! Bye,
cleaned_text = "".join([i for i in text_d if i not in string.punc
cleaned_text
```

```
1  import string
2  text_d = "Hey!!! How are you doing? And how is your health! Bye, take care."
3  cleaned_text = "".join([i for i in text_d if i not in string.punctuation])
4  cleaned_text
```

'Hey How are you doing And how is your health Bye take care'

Code 4: We will now remove the punctutions as well as convert the text to lowercase.

```
text_d = "Hey!!! How are you doing? And how is your health! Bye,
cleaned_text = "".join([i.lower() for i in text_d if i not in str
cleaned_text
```

```
1
2  text_d = "Hey!!! How are you doing? And how is your health! Bye, take care."
3  cleaned_text = "".join([i.lower() for i in text_d if i not in string.punctuation])
4  cleaned_text
```

'hey how are you doing and how is your health bye take care'

Code 5: We will now use a standard `nltk` library. Tokenization will be done here using NLTK library. The output is also pasted below.

```
import nltk
text_d = "Hey!!! How are you doing? And how is your health! Bye,
nltk.tokenize.word_tokenize(text_d)
```

```
1  import nltk
2  text_d = "Hey!!! How are you doing? And how is your health! Bye, take care."
3  nltk.tokenize.word_tokenize(text_d)
```

```
['Hey',
 '!',
 '!',
 '!',
 'How',
 'are',
 'you',
 'doing',
 '?',
 'And',
 'how',
 'is',
 'your',
 'health',
 '!',
 'Bye',
 ',',
 'take',
 'care',
 '.']
```

Note that in the output of the code, we have all the words including the punctuation marks as different tokens. If you wish to exclude the punctuations, you can clean the punctuations using the code snippets shared earlier.

Code 6: Next comes the stop words. We will remove the stopwords using nltk library. Post that, we are tokenizing the words.

```
stopwords = nltk.corpus.stopwords.words('english')
text_d = "Hey!!! How are you doing? And how is your health! Bye,
text_new = "".join([i for i in text_d if i not in string.punctuat
print(text_new)
words = nltk.tokenize.word_tokenize(text_new)
print(words)
words_new = [i for i in words if i not in stopwords]
print(words_new)
```

```
1  stopwords = nltk.corpus.stopwords.words('english')
2  text_d = "Hey!!! How are you doing? And how is your health! Bye, take care."
3  text_new = "".join([i for i in text_d if i not in string.punctuation])
4  print(text_new)
5  words = nltk.tokenize.word_tokenize(text_new)
6  print(words)
7  words_new = [i for i in words if i not in stopwords]
8  print(words_new)
```

```
Hey How are you doing And how is your health Bye take care
['Hey', 'How', 'are', 'you', 'doing', 'And', 'how', 'is', 'your', 'health', 'Bye', 'take', 'care']
['Hey', 'How', 'And', 'health', 'Bye', 'take', 'care']
```

Code 7: We will now perform stemming on a text example. We use NLTK library for it. The words are first tokenized and then we apply stemming on

them.

```
import nltk
from nltk.stem import PorterStemmer
stem = PorterStemmer()
text = "eats eating studies study"
tokenization = nltk.word_tokenize(text)
for word in tokenization:
    print("Stem for {} is {}".format(word, stem.stem(w)))
```

```
1  import nltk
2  from nltk.stem import PorterStemmer
3  stem = PorterStemmer()
4  text = "eats eating studies study"
5  tokenization = nltk.word_tokenize(text)
6  for word in tokenization:
7      print("Stem for {} is {}".format(word, stem.stem(w)))
```

```
Stem for eats is studi
Stem for eating is studi
Stem for studies is studi
Stem for study is studi
```

Code 8: We will now perform lemmatization on a text example. We use NLTK library for it. The words are first tokenized and then we apply lemmatization on them.

```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "eats eating studies study"
tokenization = nltk.word_tokenize(text)
for word in tokenization:
    print("Lemma for {} is {}".format(word, wordnet_lemmatizer.le
```

```
1  import nltk
2  from nltk.stem import WordNetLemmatizer
3  wordnet_lemmatizer = WordNetLemmatizer()
4  text = "eats eating studies study"
5  tokenization = nltk.word_tokenize(text)
6  for word in tokenization:
7      print("Lemma for {} is {}".format(word, wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma for eats is study
Lemma for eating is study
Lemma for studies is study
Lemma for study is study
```

Observe and compare the difference between the two outputs of stemming
and lemmatization. For studies and studying, stemming generated the output
at studi while lemmatization generated correct output as study.

We covered bag-of-words, tf-idf and N-gram approaches so far. But in all of
these techniques, the relationship between words has been neglected which is
used in *word embeddings* – our next topic.

Word Embeddings

"A word is characterized by the company it keeps" – John Rupert Firth.

So far, we studied a number of approaches, but all the techniques ignore the
contextual relationship between words. Let's study by an example.

Imagine we have 100,000 words in our vocabulary – starting from aa to
zoom. Now, if we perform one-hot encoding we studied in the last section, all
of these words can be represented in a vector form. Each word will have a
unique vector. For example, if the position of the word king in 21000, the
vector will have shape like the vector below which has 1 at the 21000
position and rest of the values as 0.

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0....................1,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

There are a few glaring issues with this approach:

a. The number of dimensions is very high to compute and complex.
b. The data is very sparse in nature.
c. If n new words have to be entered, the vocabulary increases by n and hence each vector dimensionality increases by n.
d. This approach ignores the relationship between words. We know that ruler, king, monarch is sometimes used interchangeably. In the one-hot-encoding approach, any such relationships are ignored.

If we wish to perform language translation, or generate a chat bot, we need to pass such knowledge to the machine learning solution. Word embeddings provide a solution to the problem. They convert the high-dimensional word features into lower dimensions while maintaining the contextual relationship. Word-embeddings allow us to create much more generalized models. We can understand the meaning by looking at an example.

In the example shown below in Figure 7-7, the relation of "man" to "woman" is similar to "king" to "queen", "eat" to "eating" is similar as "study" to "studying" or "UK" to "London" is similar to "Japan" to "Tokyo".

Figure 7-7 Word embeddings can be used to represent the relationships between words. For example, there is a relation from men to women which is similar to king to queen.
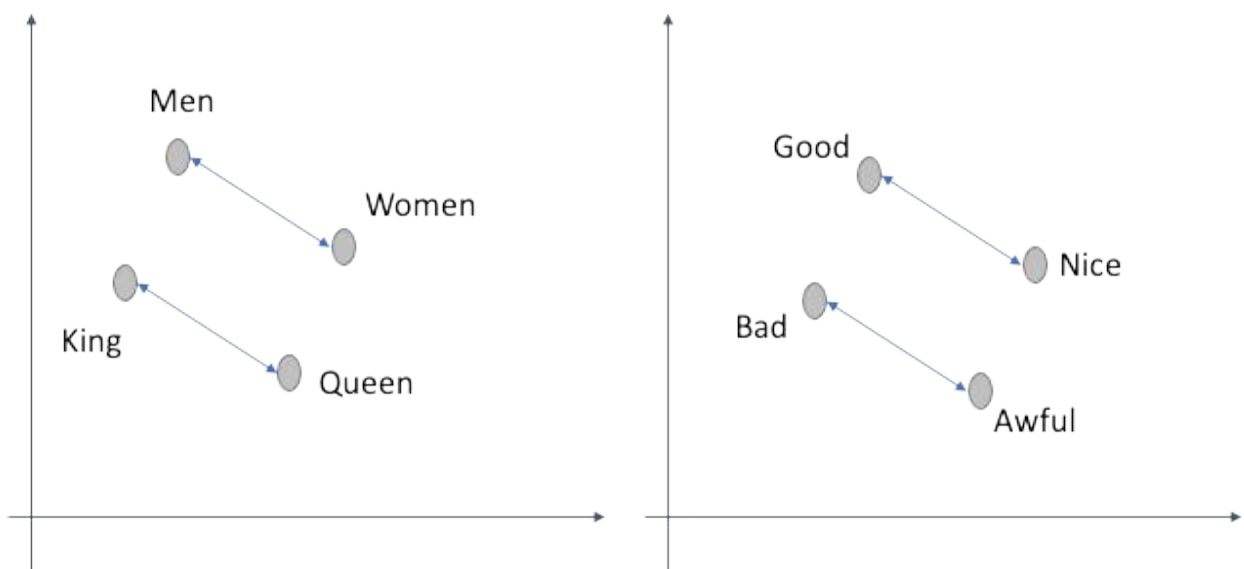
In simple terms, using word embeddings we can represent the words similarly which have similar meaning. Word embeddings can be thought as a class of techniques where we represent each of the individual words in a predefined vector space. Each of the word in the corpus is mapped to one vector. The distributed representation is understood based on the word's usage. Hence, words which can be used similarly have similar representations. This allows the solution to capture the underlying meaning of the words and their relationships. Hence, the meaning of the word plays a significant role. This representation hence is more intelligent as compared to bag of words approach where each word is treated differently, irrespective of their usage. Also, the number of dimensions is lesser as compared to one-hot encoding. Each word is represented by 10 or 100s of dimensions which is significantly less than one-hot encoding approach where more than 1000s of dimensions are used for representation.

We will cover two most popular techniques Word2Vec and GloVe in the next section. The section provides an understanding of Word2Vec and GloVe. The mathematical foundation for Word2Vec and GloVe are beyond the scope of this book. We are providing un understanding on the working mechanism of the solutions and then developing Python code using Word2Vec and GloVe. There are a few terms which we have not discussed in the book so far, so the next section on Word2Vec and GloVe might be quite tedious to understand. If you are interested only in the application of the solutions, you can skip the next section.

Word2Vec and GloVe

Word2Vec was first published in 2013. It was developed by Tomas Mikolov, et al. at Google. We are sharing the link to the paper at the end of the chapter. You are advised to study the paper thoroughly.

Word2Vec is a group of models used to produce word embeddings. The input is a large corpus of text. The output is a vector space, with a very large number of dimensions. In this output, each of the word in the corpus is assigned a unique and corresponding vector. The most important point is that the words which have similar or common context in the corpus, are located similar in the vector space produced.

In Word2Vec, the researchers introduced two different learning models – Continuous Bag of Words and Continuous Skip-gram model, which we are covering briefly:

a. Continuous bag of words or CBOW: in CBOW, the model makes a prediction of the current word from a window of surrounding context words. So, the CBOW learns Recall that in bag of words approach, the order of the words does not play any part. Similarly, in CBOW, the order of the words is insignificant.
b. Continuous skip-gram model: it uses the current word to predict the surrounding window of context words. While doing so, it allocates more weight to the neighboring words as compared to the distant words.

GloVe or Global vectors for Word Representation is an unsupervised learning algorithm for generating vector representation for words. It was developed by Pennington, et al. at Stanford and launched in 2014. It is a combination of two techniques – matrix factorization techniques and local context-based learning used in Word2Vec. GloVe can be used to find relationships like zip codes and cities, synonyms etc. It generated a single set of vectors for the words having the same morphological structure.

Both the models (Word2Vec and GloVe) learn and understand vector representation of their words from the co-occurrence information. Co-occurrence means how frequently the words are appearing together in a large corpus. The prime difference is that word2vec is a prediction-based model, while GloVe is a frequency-based model. Word2Vec predicts the context given a word while GloVe learns the context by creating a co-occurrence matrix on how frequent a word appears in a given context.

☐

POP QUIZ – answer these question to check your understanding.. Answers at the end of the book

(1)   BOW is more rigorous than tf-idf approach. True or False.

(2)   Differentiate between Word2Vec and GloVe.

We will now move to the Case Study and Python implementation in the next section.

Sentiment analysis case study with Python implementation

We have so far discussed a lot of concepts on NLP and Text data. In this section, we are first going to explore a business case and then develop Python solution on the same. We are working on Sentiment analysis.

Product reviews are a rich source of information – both to the customers and the organizations. Whenever we wish to buy any new product or services, we tend to look at the reviews by fellow customers. You might have reviewed products and services yourself. These reviews are available at Amazon, blogs, surveys etc.

Let's consider a case. A water utilities provider receives complaints from its customers, reviews about the supply and comments about the overall experience. The streams can be – product quality, pricing, onboarding experience, ease of registration, payment process, supply reviews, power reviews etc. We want to determine the general context of the review – whether it is positive, negative or neutral. The reviews have the number of stars allocated, actual text reviews, pros and cons about the product/service, attributes etc. But at the same time, there are a few business problems like:

1. Many times, it is observed that the numbers of stars received ay a product/service is very high, while the actual reviews are quite negative.
2. The organizations and the product owners, need to know which features are appreciated by the customers and which features are disliked by the customers. The team can then work on improving the features which are disliked by the customers.
3. There is also a need to gauge and keep an eye on the competition! The organizations need to know, the attributes of the popular products of their competitors.
4. The product owners can better plan for the upcoming features they wish to release in the future.

So, the business teams will be able to answer these two most important

questions:

a. What is our customer's satisfaction levels for the products and services?
b. What are the major pain points and dissatisfactions of the customers, what drives the customers engagement, which services are complex and time-consuming, and which are the most liked services/products?

This business use case will drive the following business benefits:

1. The products and services which are most satisfactory and are most liked ones should be continued.
2. The ones which are not liked and receiving a negative score have to be improved and challenges have to be mitigated.
3. The respective teams like finance, operations, complaints, CRM etc. can be notified and they can work individually to improve the customer experience.
4. The precise reasons of liking or disliking the services will be useful for the respective teams to work in the correct direction.
5. Overall, it will provide a benchmark to measure the Net Promoter Score (NPS) score for the customer base. The business can strive to enhance the overall customer experience.
6. We might like to represent these findings by means of a dashboard. This dashboard will be refreshed at a regular cycle like monthly or quarterly refresh.

To solve this business problem, the teams can collect relevant data from websites, surveys, Amazon, blogs etc. And then an analysis can be done on that dataset. It is relatively easy to analyze the structured data. In this example we are going to work on text data.

The Python Jupyter notebook is checkedin at the Github location. You are advised to use the Jupyter notebook from the GitHub location as it contains more steps.

Step 1: We import all the libraries here.

```
#### Loading all the required libraries here
from lxml import html
import requests
```

```
import pandas as pd
from nltk.corpus import stopwords
from textblob import TextBlob
import matplotlib.pyplot as plt
import sys
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import sklearn
import scikitplot as skplt
import nltk
#to ignore warnings
import warnings
warnings.filterwarnings("ignore")
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

Step 2: We will define the tags here. These tags are used to get the attributes of the product from the reviews.

```
xpath_reviews = '//div[@data-hook="review"]'
reviews = parser.xpath(xpath_reviews)
xpath_rating  = './/i[@data-hook="review-star-rating"]//text()'
xpath_title   = './/a[@data-hook="review-title"]//text()'
xpath_author  = './/a[@data-hook="review-author"]//text()'
xpath_date    = './/span[@data-hook="review-date"]//text()'
xpath_body    = './/span[@data-hook="review-body"]//text()'
xpath_helpful = './/span[@data-hook="helpful-vote-statement"]//te
```

Step 3: We are now making everything ready to extract the data. We are creating a dataframe to store the customer reviews. Then we are iterating through all the reviews and then extracting the information.

```
# Create a dataframe here.

reviews_df = pd.DataFrame()
for review in reviews:
    rating  = review.xpath(xpath_rating)
    title   = review.xpath(xpath_title)
    author  = review.xpath(xpath_author)
    date    = review.xpath(xpath_date)
    body    = review.xpath(xpath_body)
    helpful = review.xpath(xpath_helpful)
```

```
    review_dict = {'rating': rating,
                   'title': title,
                   'author': author,
                   'date': date,
                   'body': body,
                   'helpful': helpful}
    reviews_df = reviews_df.append(review_dict, ignore_index=True
all_reviews = pd.DataFrame()
```

Step 4: Let's iterate through the reviews and then fill the details.

```
# Fill the values of the reviews here. .

for i in range(1,90):
    amazon_url = 'https://www.amazon.co.uk/Hive-Heating-Thermosta
    headers = {'User-Agent': user_agent}
    page = requests.get(amazon_url, headers = headers)
    parser = html.fromstring(page.content)
    xpath_reviews = '//div[@data-hook="review"]'
    reviews = parser.xpath(xpath_reviews)
    reviews_df = pd.DataFrame()
    xpath_rating  = './/i[@data-hook="review-star-rating"]//text(
    xpath_title   = './/a[@data-hook="review-title"]//text()'
    xpath_author  = './/a[@data-hook="review-author"]//text()'
    xpath_date    = './/span[@data-hook="review-date"]//text()'
    xpath_body    = './/span[@data-hook="review-body"]//text()'
    xpath_helpful = './/span[@data-hook="helpful-vote-statement"]
    #print(i)
    for review in reviews:
        rating  = review.xpath(xpath_rating)
        title   = review.xpath(xpath_title)
        author  = review.xpath(xpath_author)
        date    = review.xpath(xpath_date)
        body    = review.xpath(xpath_body)
        helpful = review.xpath(xpath_helpful)

        review_dict = {'rating': rating,
                       'title': title,
                       'author': author,
                       'date': date,
                       'body': body,
                       'helpful': helpful}
        reviews_df = reviews_df.append(review_dict, ignore_index=
    #print(reviews_df)
    all_reviews = all_reviews.append(reviews_df)
```

Step 5: Let's have a look at the output we generated.

```
all_reviews.head()
```

Step 6: we will now save the output to a path. You can give your own path.

```
out_folder = '/Users/vaibhavverdhan/Book/UnsupervisedLearningBook
all_reviews.to_csv(out_folder + 'Reviews.csv')
```

Step 7: Load the data and analyze it.

```
#Load the data now and analyse it
data_path = '/Users/vaibhavverdhan/Book/UnsupervisedLearningBookF
reviewDataCSV = 'Reviews.csv'
reviewData = (pd.read_csv(data_path+reviewDataCSV,index_col=0,))
```

Step 8: We will now look at the basic information about the dataset

```
reviewData.shape
reviewData.rating.unique()
reviewData.rating.value_counts()
```

Step 9: We will now look at the distribution of the stars given in the reviews. This will allow us to understand the reviews given by the customers.

```
labels = '5 Stars', '1 Star', '4 Stars', '3 Stars', '2 Stars'
sizes = [reviewData.rating.value_counts()[0], reviewData.rating.v
colors = ['green', 'yellowgreen', 'coral', 'lightblue', 'grey']
explode = (0, 0, 0, 0, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```

Step 10: Make the text as lowercase, and then we remove the stop words and the words which have highest frequency.

```
reviewData.body = reviewData.body.str.lower()
reviewData.body = reviewData.body.str.replace('[^\w\s]','')
stop = stopwords.words('english')
reviewData.body = reviewData.body.apply(lambda x: " ".join(x for
freq = list(freq.index)
reviewData.body = reviewData.body.apply(lambda x: " ".join(x for
```

```
freq = pd.Series(' '.join(reviewData.body).split()).value_counts(
freq = list(freq.index)
reviewData.body = reviewData.body.apply(lambda x: " ".join(x for
```

Step 11: tokenize the data now.

```
from nltk.tokenize import word_tokenize
tokens = word_tokenize(reviewData.iloc[1,1])
print(tokens)
```

Step 12: we are performing lemmatization now.

```
from textblob import Word
reviewData.body = reviewData.body.apply(lambda x: " ".join([Word(
reviewData.body.head()
```

Step 13: Now we are appending all the reviews to the string.

```
sentimentString = reviewData.iloc[1,1]
# append to this string
for i in range(2,len(reviewData)):
    sentimentString = sentimentString + reviewData.iloc[i,1]
```

Step 14: the sentiment analysis is done here. From TextBlob we are taking the sentiment method. It generates polarity and subjectivity for a sentiment.

```
# the functions generates polarity and subjectivity here, subsett
allReviewsSentiment = reviewData.body[:900].apply(lambda x: TextB
# this contains boths subjectivity and polarity
allReviewsSentimentComplete = reviewData.body[:900].apply(lambda
allReviewsSentimentComplete.head()
```

Step 15: save the sentiment to a csv file.

```
allReviewsSentiment.to_csv(out_folder + 'ReviewsSentiment.csv')
```

Step 15: we will now allocate a meaning or a tag to the sentiment. We are classifying each of the score from extremely satisfied to extremely dissatisfied.

```
allReviewsSentimentDF = allReviewsSentiment.to_frame()
# Create a list to store the data
grades = []
```

```python
# For each row in the column,
for row in allReviewsSentimentDF['body']:
    # if more than a value,
    if row >= 0.75:
        grades.append('Extremely Satisfied')
    elif (row >= 0.5) & (row < 0.75):
        grades.append('Satisfied')
    elif (row >= 0.2) & (row < 0.5):
        grades.append('Nice')
    elif (row >= -0.2) & (row < 0.2):
        grades.append('Neutral')
    elif (row > -0.5) & (row <= -0.2):
        grades.append('Bad')
    elif (row >= -0.75) & (row < -0.5):
        grades.append('Dis-satisfied')
    elif  row < -0.75:
        grades.append('Extremely Dis-satisfied')
    else:
        # Append a failing grade
        grades.append('No Sentiment')

# Create a column from the list
allReviewsSentimentDF['SentimentScore'] = grades
allReviewsSentimentDF.head()
```

Step 16: We will now look at the sentiment scores and plot them too. Finally, we will merge them with the main dataset.

```python
allReviewsSentimentDF.SentimentScore.value_counts()
allReviewsSentimentDF['SentimentScore'].value_counts().plot(kind=
#### Merge the review data with Sentiment generated

# add column Polarity Score
reviewData['polarityScore'] = allReviewsSentimentDF['body']
```

In this case study, you not only scraped the reviews from the website, you also analyzed the dataset. If we compare the sentiments, we can see that the stars given to a product, do not represent a true picture.

In (), we are comparing the actual stars and the output from sentiment analysis. We can observe that 73% have given 5 stars and 7% have given 4 stars, while in the sentiment analysis most of the reviews have been classified as neutral. This is the real power of sentiment analysis!

Figure 7-8 Compare the original distribution of number of stars on the left side, observe the real results from the sentiment analysis

```
1
2  labels = '5 Stars', '1 Star', '4 Stars', '3 Stars', '2 Stars'
3  sizes = [reviewData.rating.value_counts()[0], reviewData.rating.value_counts()[1],reviewD
4  colors = ['green', 'yellowgreen', 'coral', 'lightblue', 'grey']
5  explode = (0, 0, 0, 0, 0)  # explode 1st slice
6
7  # Plot
8  plt.pie(sizes, explode=explode, labels=labels, colors=colors,
9          autopct='%1.1f%%', shadow=True, startangle=140)
10
11 plt.axis('equal')
12 plt.show()
```

```
1  allReviewsSentimentDF['SentimentScore'].value_counts().plot(kind='bar')
```
<AxesSubplot:>

Sentiment analysis is quite an important use case. It is very useful for the business and the product teams.

We will now move to the second case studt on document classification using Python.

Text clustering using Python

Consider this. You have got a bunch of text datasets or documents. But they all are mixed up. We do not know the text belongs to which class. In this case, we will assume that we have two types of text datasets with us – one which has all the data related to football and second class is travel. We will develop a model which can segregate these two classes.

Step 1: Import all the libraries

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import numpy as np
import pandas as pd
```

Step 2: We are now creating a dummy dataset. This text data has a few sentences we have written ourselves. There are two categories -

```
text = ["It is a good place to travel",
        "Football is a nice game", "Lets go for holidays and
        "It is a goal, a great game.", "Enjoy your journey an
```

Step 3: We will use tfidf to vectorize the data.

```
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
X = tfidf_vectorizer.fit_transform(text)
```

Step 4: let's do the clustering now.

```
k = 2
model = KMeans(n_clusters=k, init='k-means++', max_iter=10, n_ini
model.fit(X)
```

Step 5: Lets represent centroids and print the outputs.

```
centroids = model.cluster_centers_.argsort()[:, ::-1]
features = vectorizer.get_feature_names()

for i in range(k):
    print("Cluster %d:" % i),
    for ind in centroids[i, :10]:
        print("%s" % terms[ind])
```

```
1  for i in range(k):
2      print("Cluster %d:" % i),
3      for ind in centroids[i, :10]:
4          print("%s" % terms[ind])
```

Cluster 0:
travel
ready
teams
good
place
lets
holidays
egypt
journey
rest
Cluster 1:
game
great
football
nice
goal
travel
enjoy
fortget
good
holidays

You can extend this example to other datasets too. Use your own dataset and replicate the code in the example above.

There is no more Python Jupyter notebook which uses Word2Vec and GloVe. We have checked-in the code at the GitHub location of the book. You are advised to use it. It is really an important source to represent text data.

With this, we are coming to the end of this exciting chapter. Let's move to the summary now.

7.6 Summary

Text data is one of the most useful datasets. A lot of intelligence is hidden in the texts. Logs, blogs, reviews, posts, tweets, complaints, comments, articles and so on – the sources of text data are many. Organizations have started to invest in setting up the infrastructure for accessing text data and storing it. Analyzing text data requires better processing powers and better machines. It requires special skill sets and deeper understanding of the concepts. NLP is an evolving field and a lot of research is underway. At the same time, we cannot ignore the importance of sound business acumen and knowledge.

Data analysis and machine learning are not easy. We have to understand a lot of concepts around data cleaning, exploration, representation and modelling. But, analyzing unstructured data might be even more complex than structured datasets. We worked on images dataset in the last chapter. In the current chapter, we worked on text data.

Text data is one of the most difficult to analyze. There are so many permutations and combinations for text data. Cleaning the text data is not easy and is quite a complex task. In this chapter we discussed few of those important techniques to clean the text data. We also covered few important methods to represent text data in vector forms. You are advised to do practice of each of these methods and compare the performances by applying each of the techniques.

With this, we come to an end to the chapter seven of the book. This also

marks an end to part two of the book. In the next part of the book, the complexity increases. We will be studying even deeper concepts of unsupervised learning algorithms.

You can now move to the practice questions.

Practical next steps and suggested readings

1. Get the datasets from the link below. You will find a lot of text datasets here. You are advised to implement clustering and dimensionality reduction solutions.
    https://blog.cambridgespark.com/50-free-machine-learning-datasets-natural-language-processing-d88fb9c5c8da
2. This is the second source of text datasets, where you will find a lot of useful datasets.
    https://www.kaggle.com/datasets?search=text
3. You are advised to go through the research paper – Efficient Estimation of Word Representations in Vector Space by Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean
    https://arxiv.org/pdf/1301.3781.pdf
4. You are advised to go through the research paper – GloVe: Global Vectors for Word Representation by Jeffrey Pennington, Richard Socher, Christopher D. Manning
    https://nlp.stanford.edu/pubs/glove.pdf
5. There are a few papers which are really quoted widely
6. Avrim Blum and Tom Mitchell: Combining Labeled and Unlabeled Data with Co-Training, 1998
7. Kevin Knight: Bayesian Inference with Tears, 2009.
8. Thomas Hofmann: Probabilistic Latent Semantic Indexing, SIGIR 1999.
9. Donald Hindle and Mats Rooth. Structural Ambiguity and Lexical Relations, Computational Linguistics, 1993.
10. Collins and Singer: Unsupervised Models for Named Entity Classification, EMNLP 1999.
11. You are advised to go through the research paper- Using TF-IDF to Determine Word Relevance in Document Queries by Juan Ramos
    https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424&rep=rep1&type=pdf

12. You are advised to go through the research paper – An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation by Bijoyan das and Sarit Chakraborty
https://arxiv.org/pdf/1806.06407.pdf